AUTOMATED DAMAGE ASSESSMENT OF REINFORCED CONCRETE COLUMNS FOR POST-EARTHQUAKE EVALUATIONS

A Dissertation
Presented to
The Academic Faculty

By

Stephanie Ann German

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy in the
School of Civil and Environmental Engineering

Georgia Institute of Technology
March, 2013

AUTOMATED DAMAGE ASSESSMENT OF REINFORCED CONCRETE COLUMNS

FOR POST-EARTHQUAKE EVALUATIONS

Approved by:

Dr. Reginald DesRoches, Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Ioannis Brilakis, Co-Advisor
School of Civil and Environmental Engineering
University of Cambridge

Dr. Ghassan AlRegib
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Laura Lowes
School of Civil and Environmental Engineering
University of Washington

Dr. Yang Wang
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date Approved: [March 25, 2013]

ACKNOWLEDGEMENTS

First and foremost, I offer my sincere gratitude to my co-advisors, Dr. Reginald DesRoches and Dr. Ioannis Brilakis. They have supported me during my Doctoral studies by imparting their own knowledge to me as well as by being patient and allowing me the room to work and learn on my own.

I am honored to have Dr. Laura Lowes, Dr. Yang Wang and Dr. Ghassan AlRegib as members of my dissertation committee. Their expertise in each of their individual fields has proven to broaden my understanding of my own research as well as inspire me towards new areas and ideas for the future. I gratefully acknowledge them for their valuable guidance and encouragement along the way.

I am very grateful for the wonderful other graduate students which I have had the opportunity to meet and collaborate with during my time at Georgia Tech. Without my current and former office mates from CITL to Mason 522A to IPST, I would not have enjoyed my time in the office anywhere near as much. They include Dr. Zhenhua Zhu, Dr. Christian Koch, Mr. Habib Fatih, Mr. Abbas Rashidi, Ms. Atefe Makhmalbaf, Ms. Gauri Jog, Ms. Aswathy Sivaram, Mr. ManWoo Park, Dr. Fei Dai, Dr. Abdollah Shafieezadeh, Dr. Karthik Ramanathan, Dr. Jieun Hur, Mr. Jong-Su Jeon, Mr. Timothy Wright, Mr. Nathan Mayercsik, Mrs. Laura Redmond, Mr. Pablo Vega Behar and Mr. Dapeng Zhu. I am extremely grateful to Dr. Zhenhua Zhu who preceded me on this research project and was patient and kind enough during my first few years as a Doctoral student to encourage and teach me. It was always a pleasure working

with you. I would also like to thank Mr. Jong-Su Jeon. It has been a joy working alongside you—writing research papers and problem-solving as a team.

I have also been very fortunate to be surrounded by very loving and encouraging friends outside of life at Georgia Tech. I am grateful to each of my good friends from Atlanta Westside. Without the families who have taken me in as if I was in fact a part of their family as well as my roommates and neighbors, I would not have enjoyed these last four years very much at all. Thank you to my best friends who never understood what I was doing, but encouraged me all the same.

I am beyond thankful to my family. Their support all throughout the last four years has driven me to this final point of success. I am thankful for my mother, who knew from the start that I would be an engineer, for encouraging me when I didn't think I wanted to be one, and to my father, whose confidence in my ability to do things actually made me believe I could do them. Of course, I owe at least 50% of the laughter I have experienced during my Doctoral studies to my three sisters. Thank you, each one of you, for being an intelligent woman yourself and so inspiring me to continue on towards this goal. Finally, I would like to thank my fiancé, Jeff Paal, for his constant support, love and thoughtfulness throughout the rigors of a Doctoral degree.

Also, I would like to thank the agency which funded my graduate studies and this work. The material presented in this research is based upon work supported by the National Science Foundation under Grant Numbers CMMI-1034845 and CMMI-0738417. Any opinions, findings and conclusions or recommendations expressed in

this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

When an earthquake occurs, entry into damage buildings as soon as possible is often necessary to expedite the response and recovery process. However, current manual post-earthquake building evaluation practices are time consuming, and the subjective results can lead to erroneous judgments. Therefore, the concept of automating these current manual practices is proposed in order to overcome these existing limitations. There have been a number of different approaches to solving this problem in the fields of structural health monitoring, remote sensing and computer vision.

With the intention of providing a real-time, cost-effective evaluation of the damaged structural members within the building, an automated method in computer vision is proposed. Several methods in computer vision and image processing have been developed for detecting damage and defects (cracks, corrosion) as well as structural elements (columns), and the success of these methods has been verified. However, there still exists several missing links in order to provide a comprehensive assessment of the damage state of a structural member. There remains a need for detection (spalling) and property retrieval (spalling, cracks) of significant and indicative damage types on the structural element surface. In addition, the link between the structural element and the damage properties needs to be established such that the properties defined in the image axes will provide meaning to the evaluator in the real-world. Finally, the column response

mechanism and maximum drift capacities need to be correlated to the visible damage observed on the structural element in the image or video frame.

The purpose of this research is to investigate an approach to joining each of these links and provide a comprehensive assessment of the response mechanism, damage state and drift capacity of a reinforced concrete column based solely on the observed visible damage. Specifically, techniques from the fields of image processing and computer vision are employed in order to develop a set of methods capable of automatically detecting the spalled regions on the surface as well as the properties of cracking and spalled regions. Also, an analysis of the typical damage progression for specific RC column response mechanisms is performed in order to filter through the variables and provide damage indices based on visible damage at the residual condition of the column. The methods proposed in this research were implemented in a Microsoft Visual Studio .NET environment, and tested on real images of damaged columns retrieved from post-earthquake reconnaissance trips and image databases. The test results indicated that the methods could automatically detect spalled regions and retrieve the properties of spalling and cracks on RC columns in images or video frames, and further, that this retrieved information could be accurately translated to a meaningful assessment of the column's existing damage state.

CHAPTER 1

INTRODUCTION

1.1. Problem Description

Within the two year period from 2010-2012, there were over 500 earthquakes of magnitude 6.0 or higher worldwide. This led to an estimated 342,841 casualties (USGS, 2012). Earthquakes have been deemed one of the most costly natural hazards faced in our nation, posing a major threat to at least 75 million Americans in 39 states, and an estimated future loss of $5.6 billion each year (USGS, 2006). As of 2011, 10 of the top 20 world's costliest natural disasters since 1965 were earthquakes (Figure 1.1, The Economist, 2011).

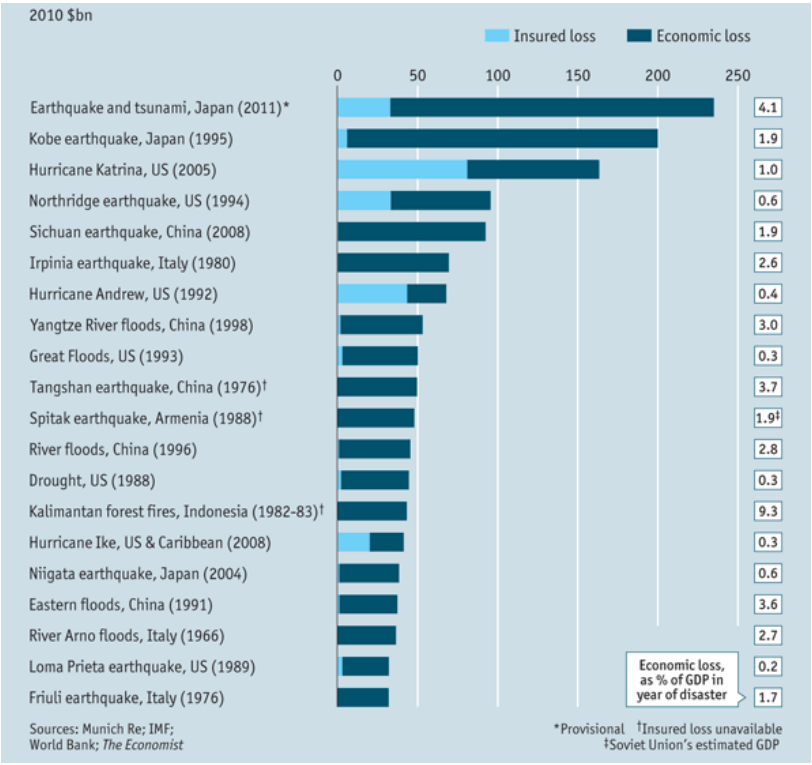


Figure 1.1: World's costliest natural disasters since 1965 (The Economist, 2011)

Furthermore, the probability of major life-threatening events occurring in the United States in the near future is significantly high. According to the results of a study performed by the Uniform California Earthquake Rupture Forecast, and partially illustrated in Figure 1.2, the probability of one or more earthquakes of magnitude 6.7 or greater capable of causing extensive damage and loss of life in California in the next 30 years is greater than 99% (Field et al., 2008). Also in Figure 1.2, the 30-year earthquake probabilities are shown for the San Francisco and Los Angeles regions individually (USGS, 2008).

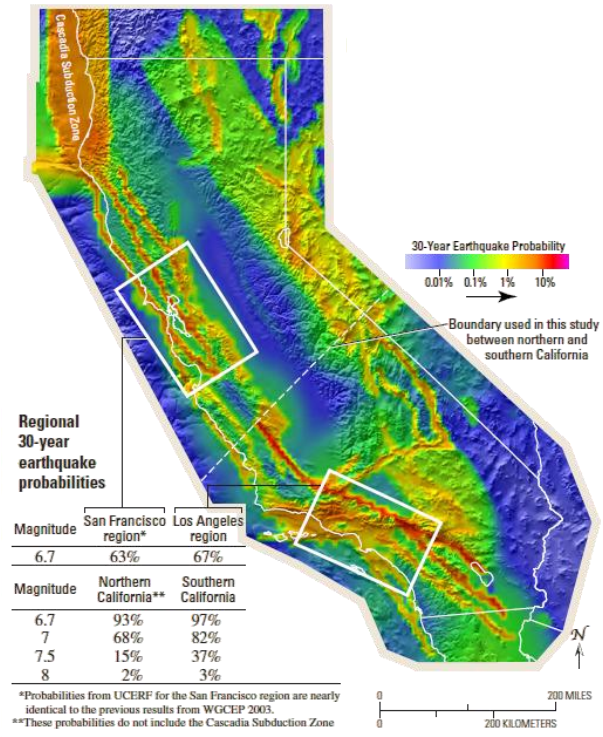


Figure 1.2: Distribution of the likelihood of an earthquake rupturing within 3-4 miles throughout the state of California (USGS, 2008)

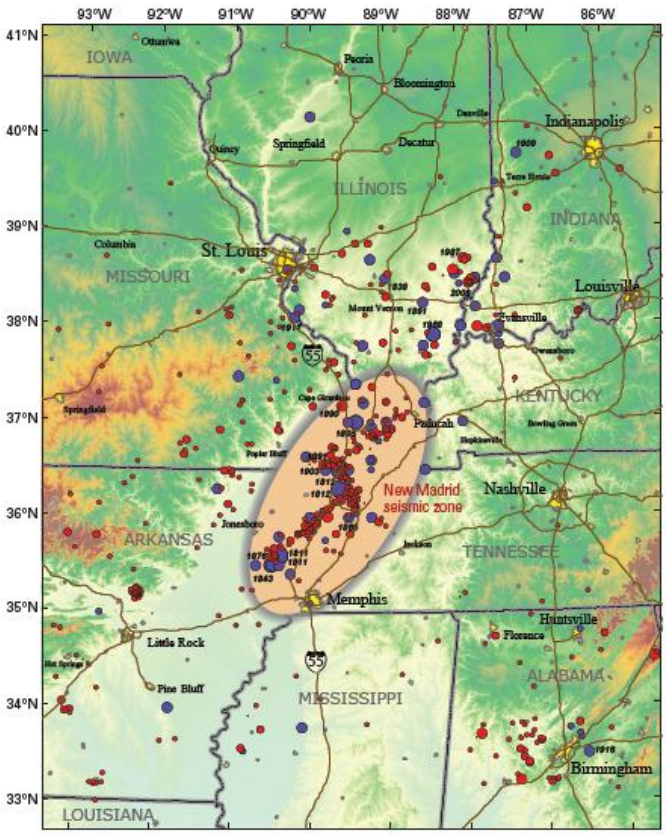


Figure 1.3: Topographic map of the central U.S. with past earthquakes greater than magnitude 2.5 (USGS, 2009)

Across the Rocky Mountains, in the central United States, the threat of significant earthquakes having a detrimental impact in the region remains. The New Madrid Seismic Zone has been frequented by earthquakes of varying magnitude for at least the last 4500 years (Figure 1.3). The last significant round of earthquakes was in 1811-1812, with estimated approximate magnitudes ranging from 7.0 to 8.0. Based on the history of this region (Figure 1.3), the chance of an earthquake similar to the 1811-1812 earthquakes occurring in the next 50 years has been estimated to be approximately 7-10%. In addition, the chance of a magnitude 6.0 or greater earthquake occurring in this region in the next 50 years is estimated to be

approximately 25-40%. An earthquake of this magnitude in this region would significantly impact large U.S. cities such as Memphis, TN, St. Louis, MO and Little Rock, AR in addition to several small and medium-sized cities due to the lack of preparedness for such an event (USGS, 2009). A scenario similar to that which may ensue in this region (the rupture of a M_W 7.7 earthquake over the entire length of each of the three segments of the New Madrid fault) was investigated in order to provide a credible assessment of the current risk to the New Madrid Seismic Zone (Elnashai et al., 2009). The scenario considered the impact due to the ground shaking as well as permanent ground deformations and secondary disasters such as fire and flooding caused by the initial disaster, and the analysis comprised three major components: hazard, inventory and fragility. According to this investigation, Tennessee, Arkansas, Missouri, Illinois, Kentucky, Indiana, Alabama and Mississippi are all impacted – the first three states being impacted the most severely. Nearly 715,000 buildings, including 130 hospitals, are damaged, and 15 major bridges are deemed unusable, throughout the eight states. In response, 1,500 search and rescue teams, requiring 42,000 personnel, are required. Then, three days after the earthquake, 7.2 million people remain displaced, 2 million people are in search of temporary shelter, and the direct economic losses for the eight states are estimated at nearly $300 billion, with indirect losses still unaccounted for, but estimated for at least twice that amount (Elnashai et al., 2009).

It has been established that the risk of earthquakes overall could be greatly mitigated by the provision of critical and timely information in the response following the disaster (USGS, 2006). Thus, guidelines for structural evaluations after

an earthquake event have been established by government agencies such as the Applied Technology Council (ATC) and the Federal Emergency Management Association (FEMA). The evaluation of buildings in the event of an earthquake typically occurs at two stages. Buildings must be evaluated prior to: (1) entry of emergency search and rescue teams; and (2) re-entry by occupants. Each of these evaluations is typically performed by a triage team of certified inspectors/engineers. These specialists follow existing procedures established by local branches of national government authorities in most cases; however, worldwide, several countries adhere to those guidelines established by U.S. based entities such as the Federal Emergency Management Agency (FEMA) (emergency response) and the Applied Technology Council (ATC) (occupancy) (FEMA, 2006; ATC, 1989). The team of specialists must make an assessment based on their experience and knowledge, coupled with visual observation of the damage inflicted on the load-bearing members of the structure.

As with many recorded cases of earthquakes, and particularly those near populated areas, in the October 15, 2006 Hawaii earthquake, the capacity of response teams was greatly exceeded by the demand. The safety evaluation processes took several weeks to complete due to the large number of buildings required to be assessed (Chock, 2007). Following the January 12, 2010 earthquake in Haiti, over 100,000 houses were destroyed and nearly 190,000 damaged in the Port-au-Prince and surrounding areas (DesRoches et al., 2011). Following the February 22, 2011 earthquake in the Canterbury/Christchurch area of New Zealand (third in a series of four major earthquakes in the area within a year), 10,000 houses

were damaged to the point of demolition (Simcox, 2011); however, the most significant damage took place in the Central Business District (CBD), home to 4,000 commercial buildings, 1,000 of which were designated to be demolished (Collins, 2011). In each of these cases, the large inventory of damaged buildings called for an unfeasibly large structural assessment task force, and in the aftermath of the earthquakes, several inefficiencies surfaced in current procedures for the post-earthquake building assessment processes. According to a U.S. Senate Report, Haiti had made little progress in rebuilding in the five months since its earthquake due to an absence of leadership, disagreements among donors and general disorganization (Katz, 2010). In the Canterbury region, it took ten days to inspect 75% of those buildings in the CBD, and over 1,300 commercial buildings were given red or yellow placards, restricting access to the entire CBD through September 2011, seven months after the third earthquake (RNZ, 2011). In each of these scenarios, the existing procedures in post-earthquake assessments are time-consuming. With the imminent nature of the completion of these assessments in order to reduce the economic and societal impact associated with the downtime after an earthquake, this is a vital issue. In addition to being time-consuming, the subjective nature of evaluating building safety and structural integrity may lead to erroneous judgments. Thus, there is a need for improvement of the existing assessment procedures such that they are rapid (real-time) and more reliable.

In order to overcome the existing limitations in post-earthquake safety and structural inspections, there have been several efforts towards developing an automated evaluation method which would provide a rapid and reliable estimate

similar in nature to that currently retrieved by certified inspectors and/or structural engineers. These efforts focus on retrieving visual data via high-resolution cameras, satellites or video cameras and translating the image data to meaningful damage information by way of various algorithms in image processing (i.e., pattern recognition, wavelet transforms, digital filtering, Fourier transforms, thresholding segmentation, edge detection, region-based segmentation, etc.). The results of these various efforts range in the amount of detail which is provided, from the city- or building-wide level (satellite imagery) to the specific damage on the specific elements. These methods have been tested in a wide-range of structures such as concrete bridges, pipes, tunnels and buildings, and the results of these tests do verify the ability of their application for detecting damage in assessment-based practices. However, in order to provide a rapid and reliable (quantitative) evaluation of the structural element which is indicative of the existing structural integrity of the member, there exist several gaps in the aforementioned research efforts. First, although some damage types have been explored to a substantial depth (i.e. cracks on concrete surfaces), critical types of damage which are indicative of the remaining structural integrity of a member (i.e. spalling on concrete surfaces) have yet to be addressed. In addition, the detected damage has not been correlated with the surface (structural member) on which it exists. This prevents the quantification of the damage in a manner which is meaningful for the structural assessment of the member and the building as a whole. Finally, the existing attempts to provide a rapid and reliable assessment of damage to structural members have yet to accomplish that overall goal. The visible damage, the extent of that visible

damage and the response mechanism of the structural member all need to be defined automatically such that the existing state of the element can be determined automatically as well. The main purpose of this research is to investigate the means to filling these existing gaps.

1.2. Research Objectives

The long term goal of this research is to make substantial improvements with respect to the speed and reliability of post-earthquake safety and structural assessment procedures. In order to address the aforementioned research problems, the creation of an automated assessment tool which utilizes computer vision to immediately determine the state and safety of the structural components throughout the building is proposed. The proposed methodology is based on the collection of video data during the structural evaluation walk-thru stages. In this data, first, every concrete column is retrieved and the damage inflicted on the structural elements is then located. The spatial properties of each damage type detected are considered collectively, so that the column's existing strength/stiffness can be approximated in the form of a damage index. In order to validate the proposed methodology, the aim of the research proposed here has been limited as described below. These constraints have been established based on frequency and criticality in post-earthquake scenarios.

- Building Type: reinforced concrete frame buildings
- Structural Elements: rectangular RC columns
- Response Mechanisms: flexure and shear

- Damage: cracks and spalling
- Damage Properties: crack width, crack length, crack orientation, crack spacing, spalled length, spalled depth

Thus, this dissertation presents the efforts to automatically perform a quantitative assessment of the post-earthquake integrity of RC columns in RC frame buildings based on the calculated and detected crack and spalling properties. With the prospective aim of this research established, the objectives specific to the work proposed here can be presented:

1) Determine a way to automatically retrieve crack and crack pattern properties including width, length, spacing and orientation with respect to the structural element detected. The results should then be compared with those from manual surveys to indicate the effectiveness.
2) Determine a way to automatically detection regions of spalling on concrete surfaces. The result should then be compared with those from manual surveys to evaluate the effectiveness.
3) Determine a way to automatically retrieve damage properties from the spalled detection map (2). The properties include the depth and length of the spalled region (extent of spall into the member and along the longitudinal axis of the member). These properties must be related to the dimension and orientation of the structural member to produce relative measurements, and the results must then be compared with those from manual surveys to indicate the effectiveness.

4) Determine a way to automatically estimate a reliable and quantitative, comprehensive damage state estimate for reinforced concrete columns. A novel set of damage state definitions based on visual damage data must be defined, and the results from sub-objectives 1-3 need to be translated such that the appropriate damage state can be automatically retrieved. These definitions are determined based on the correlation between the visual damage detected, the maximum drift of the structural member and the associated damage response mechanism.

1.3. Outline of the Dissertation

This dissertation is organized into six chapters with the following contents:

Chapter 2 provides a literature review of the current state of the practice in safety and structural assessments for post-earthquake damaged structures. This chapter concludes with a discussion of the limitations to these exiting practices.

Chapter 3 presents pertinent research efforts in the field of post-earthquake safety and structural assessment. The state of knowledge in assessment methods using Structural Health Monitoring (SHM), Remote Sensing (RS) and Computer Vision (CV), as well as that regarding the seismic performance of RC columns is presented in this chapter. The chapter concludes with the definition of the existing gaps in these knowledge areas for the purpose of this research.

Chapter 4 discusses the methods in automated damage detection and property retrieval in detail. In this chapter, the algorithms in spalled region detection and spalled property retrieval, as well as the method in crack pattern retrieval, are presented. The design of the experiments that validate these methods

is also presented. The implementation and results are discussed for each of these methods.

Chapter 5 discusses the classification model for RC column damage index estimation. The means to defining the vision-based damage states is discussed according to the link between the damage and the performance of RC columns in the post-earthquake scenario. The design of the model is outlined in detail, and the experiments performed in order to validate the model are described. The implementation and results of this experiment are discussed.

Chapter 6 presents the findings and contributions of this research. The dissertation is concluded with a summary of the research, a general set of conclusions and an outline for future research directions.

CHAPTER 2

STATE OF PRACTICE

2.1. State of Practice in Post-Earthquake Safety Assessments

Immediately after a disaster event, such as an earthquake or a hurricane, community response teams (most often trained firemen from the local fire department) are first dispatched for a drive-through assessment to identify locations of heavy damage and high potential of trapped victims (USFA, 1994; McEntire and Cope, 2004). The assessment is commonly interrupted by fires and other hazards causing immediate threat to lives. After initial assessments, due to the drastic effects in the aftermath of such disasters, local emergency response teams often call for backup from US&R task forces (or the equivalent government-established response task force in other countries). These task forces follow standard operating procedures in conjunction with the community emergency response team, which may vary according to locality and situation. When a disaster (e.g., an earthquake) affects multiple structures, the prioritization of damaged buildings before any search and rescue operations commence is required. There are extensive risks for people entering these buildings damaged by an earthquake, and any further structural collapse (likely due to aftershocks) could quickly transform these task force members from emergency responders into additional victims. Therefore, it is essential to evaluate the level of safety associated with damaged buildings in post-earthquake scenarios prior to the entry of the emergency search and response teams.

In current practice, safety evaluation of damaged buildings is designated to be performed by a team of various specialists—a technical search specialist, a structures specialist, a medical specialist, two canine search teams, a Haz-Mat specialist and two rescue specialists (emergency responders)—according to the US&R task force guidelines (FEMA, 2006). According to the Rescue Field Operations Guide (ROG), the search and rescue operations are organized into five phases: (1) assessment of the collapse area; (2) removal of all surface victims as quickly and safely as possible; (3) search and rescue of victims from accessible void spaces; (4) selected debris removal to locate and rescue victims; and (5) general debris removal—usually conducted after all known victims have been removed (FEMA, 2006). In this initial stage of reconnaissance, the emergency responders/structural specialists should document any and all apparent information regarding the building structure and surroundings including the building ID, cross section, floor plans, configuration, size, number of stories, occupancy type, collapse type, damage/void/hazard locations, victim location identification, best access for entry and any other notes which may prove useful in further emergency response, structural assessment and hazard mitigation operations. At the conclusion of this initial reconnaissance stage, the ROG specifies the safety of the structure in the manner depicted in Figure 2.1(a). Depending on the outcome of this reconnaissance phase, the process may be iterative. In all cases, ultimately the search and rescue results are designated on the building in the manner represented in Figure 2.1(b). This designation, established by the International Search and Advisory Group (INSARAG), provides a conclusive remark concerning the safety of the building for

the purpose of the structural assessment which should follow: G = GO or N = NO GO (FEMA, 2006).

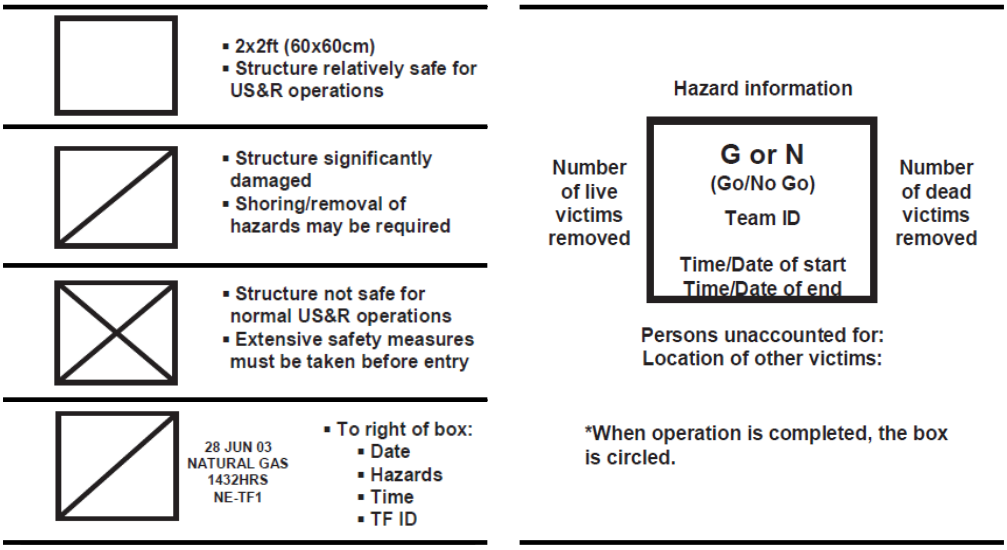


Figure 2.1: FEMA markings: (a) Structures/hazards marking for initial US&R safety assessment; (b) INSARAG structure assessment marking (FEMA, 2006)

During the safety assessment, structural specialists are expected to be the most prepared personnel to deal with all aspects of the built environment in urban areas (Aldunate et al., 2006). The involved structural specialists are responsible for identifying potential structural hazards and monitoring the structure for condition changes during the rescue and recovery operations (FEMA, 2008). FEMA suggests that two or more specialists working together, as practiced in safety/structural evaluation for occupancy (Section 2.2), is preferred and ideal, but that it is impractical in emergency response to large disasters (FEMA, 2009b). There are typically not enough qualified structural specialists for allocation to the individual

community emergency response teams. Those that can participate must be licensed professional engineers with a minimum of five years of experience, and they are required to take structural collapse technician courses and US&R structural specialist training (FEMA, 2008). In any case, collaboration-related problems, including lack of coordination, information sharing, trust and communication, between the various specialists involved in these disaster relief efforts have been identified (Kostoulas et al., 2006).

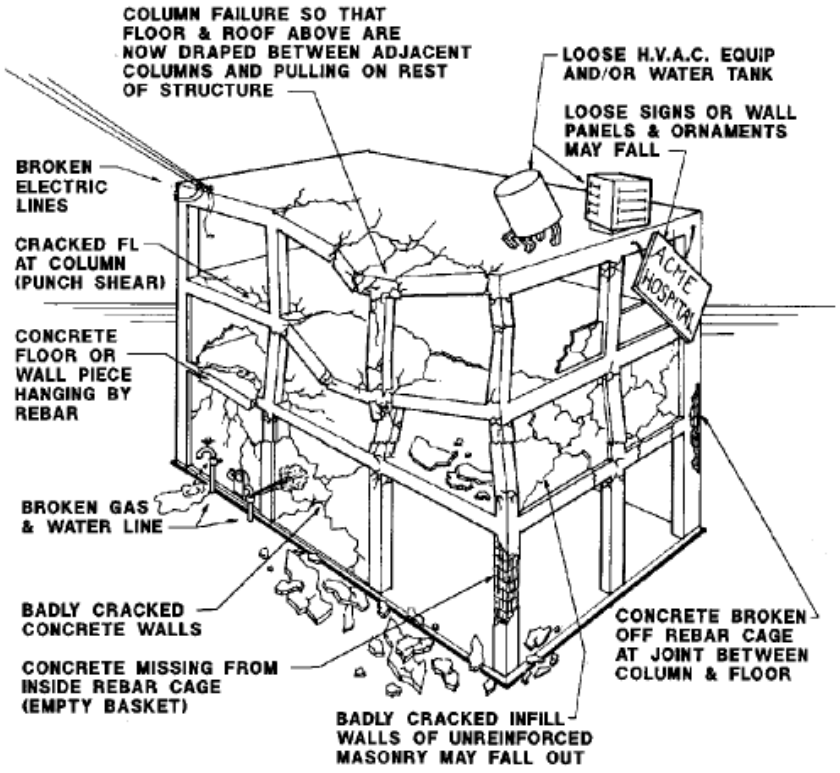


Figure 2.2: Basic collapse patterns and check points for a concrete frame building as noted in the Structural Collapse Technician Course Student Manual (FEMA, 2009a)

In order to mitigate the overwhelming demand for structural specialists and the poor communication between the specialists on the team, Search and Rescue Organizations sponsored by FEMA offer seminars and structural collapse technician courses (FEMA, 2009a; FEMA 2009b) to educate the emergency response specialists regarding the basics of structural collapse patterns and common checkpoints (i.e. Figure 2.2). However, it is difficult for emergency responders to operate rapidly to save lives while being asked to assess the building's damage state as well. As a result, from 1979-2002, the second leading activity causing firefighter fatalities due to structural collapse was participation in search and rescue operations, second only to fires (Brassell and Evans, 2003). Such collapse related injuries and deaths of emergency responders occur every year (NIOSH, 2007). Moreover, the training (seminars and technician courses) is not available to most emergency responders throughout the country (only held in certain major urban centers), and even when available, it is not free of cost to the emergency service personnel. In Texas, for example, the tuition for 50 training hours for the course on advanced structural collapse exceeds $2,000 per individual (TEEX, 2007).

Finally, in large-scale disasters, where several thousand buildings may be affected, evaluating the safety of all buildings manually by any combination of structural specialists and trained emergency response specialists requires a significant amount of time (Pena-Mora and Mehta, 2009). The Applied Technology Council (1999) suggested that emergency responders wait one to eight days before entry into damaged buildings depending on the magnitude of the earthquake main shock and the amount of time anticipated spent in the building (Table 2.1). The

adverse effect is that the survival rate for trapped victims is significantly reduced over time. Over 91% of people trapped in collapse structures can survive if they are rescued within 20 minutes; however, once trapped for two days, which is not uncommon (Table 2.1), this value declines steeply to 36.7% (UKFSSART, 2007). Thus, the existing practices in post-earthquake safety assessment for search and rescue response operations are unable to meet the demand when catastrophic disasters occur near urban areas. There is a need for more rapid and more reliable tools to assist in these safety assessment operations.

Table 2.1: Recommended wait before entry into unsafe buildings (ATC, 1999)

Main Shock Magnitude (M)	**M ≥ 6.5**	**6.0 ≤ M < 6.5**	**M < 6.0**
Enter for two hours	1 day	1 day	1 day
Enter for eight hours	3 days	2 days	1 day
Enter for 24 hours	8 days	4 days	2 days

2.2. State of Practice in Post-Earthquake Structural Assessments

Currently, evaluating the post-earthquake condition of buildings for the purpose of informing occupants whether or not they can return is the task of structural engineers and building inspectors. Usually, after all emergency operations have been concluded, reconnaissance teams with licensed inspectors are deployed to the affected areas. These teams conduct manual inspections of buildings according to guidelines, such as those outlined by the Applied Technology Council (ATC) in ATC-20 (1989) and ATC-20-2 (1995). ATC-20-2 recommends that buildings be classified

as: (1) UNSAFE, the building is an extreme hazard and may collapse and thus, is unsafe for occupancy or entry except by authorities (red-tag); (2) RESTRICTED USE (LIMITED ENTRY), detailed use and entry restrictions are specified for each particular building by the inspectors and enforced by the building owner (yellow-tag); or (3) INSPECTED (SAFE), no apparent hazard found; thus, the building is safe for entry and occupancy as the event has not significantly affected the structural integrity of the building (green-tag). With respect to concrete frame buildings ATC-20 (1989) and ATC-20-2 (1995) provide recommendations for accomplishing the classification, based primarily on observed column damage. For example, a cast-in-place concrete building is regarded as unsafe when any of three conditions exists in the columns: (1) buckled or fractured reinforcement; (2) massive concrete spalling and exposure of longitudinal reinforcement; or (3) wide diagonal cracks through the column depth (ATC, 1989).

The system employed by these guidelines consists of three tiers of evaluation: “Rapid,” “Detailed” and “Engineering” (Table 2.2). Ideally, within the first few hours or days after the earthquake, evaluation of all buildings in the damaged area undergo the “Rapid Evaluation” procedure, which is largely focused on inspection of the exterior of the structure, noting a general level of damage or suspected damage areas. The building is entered only when sufficient view of the structure is obstructed from the outside, or when some sort of problem such as gross nonstructural distress (e.g., fallen ceiling or badly damaged partition walls) is presumed (ATC, 1995). Thus, the “Rapid Evaluation” is designed for application to those structures which can quickly and obviously be posted as either SAFE or

UNSAFE. After this phase of inspection, all remaining structures are considered LIMITED ENTRY and will undergo a "Detailed Evaluation." Here, a detailed visual examination of all questionable portions of the structure is performed by a group of two structural engineers (preferably).

Table 2.2: ATC-20 stages of building evaluation (ATC, 1995)

Technique	Required Personnel	Goal	Example Time per Building
Rapid Evaluation	• Qualified building inspectors • Civil/structural engineers • Architects • Other individuals deemed qualified by local jurisdiction	• Rapid assessment of safety. • Used to quickly post obviously unsafe and safe structures, and to identify buildings requiring Detailed Evaluation	10-20 minutes
Detailed Evaluation	• Structural Engineers*	• Careful visual evaluation of damaged buildings and questionable situations. • Used to identify buildings requiring an Engineering Evaluation.	1-4 hours
Engineering Evaluation	• Structural engineering consultant*	• Detailed engineering investigation of damaged buildings, involving use of construction drawings, damage data and new structural calculations.	1-7 days or more

* Geotechnical specialists required for assessment of geotechnical hazards

The “Detailed Evaluation” entails an assessment of the following: (1) overall damage to the structure, (i.e., specification of collapse/partial collapse, noticeably leaning building/individual story, fractured foundations); (2) vertical load system—local (column buckling) and global failure (partial collapse) are considered; (3) lateral load system—severely cracked shear walls, hinging and massive spalling in RC columns, highly damaged diaphragms; (4) P-Delta effects—residual story drift; (5) degradation of the structural system—cracking, spalling and local crushing of RC frame members; (6) falling hazards; and (7) slope or foundation distress (ATC, 1995). In each of the items in this inspection list, the means to defining the safety posting is qualitative. Through this assessment, the building is again posted as either SAFE or UNSAFE.

Those buildings which still do not fall into either of these two categories are subjected to an “Engineering Evaluation.” Such an evaluation is the last and most rigorous level of building assessment, and is only recommended after it has been determined that the building has incurred damage to such a degree that visual inspection techniques are no longer sufficient to assess the condition of the structure. Each engineering evaluation takes several days to weeks to complete, and requires a structural engineering consultant to perform the inspection. At this time, the engineer considers the construction drawings, damage data and structural calculations in addition to the visual investigation of the damaged building (ATC, 1995). Although the methods outlined in these documents (ATC, 1989; ATC, 1995) do achieve the desired level of building condition evaluation, it is generally difficult or impossible to post every building in the damaged area as either SAFE or UNSAFE

quickly after the event due to the lack of qualified inspectors, the amount of work associated with each level of inspection and the sheer amount of inspections required (ATC, 1989).

Table 2.3: Draft SEAONC Disaster Emergency Services Committee building assessment criteria (ATC, 1989)

Building Element	Type of Damage		
	RED (5-6)	**YELLOW (3-4)**	**GREEN (1-2)**
Columns	• Significantly out of plumb • Moderate buckling • Severe cracking • Signs of yielding • Falling hazard • Damaged, so as to no longer be providing support to level above	• No apparent instability hazard • Minor cracking	
Foundations	• Moderate foundation damage • Severe building settlement • Building has slid off its foundation and super-structure is in jeopardy	• No apparent instability hazard • Moderate foundation cracking for buildings on grade • Moderate building settlement	• Minor building settlement • Minor foundation cracking • Evidence of local uplift
Parapet Walls	• Partial collapse	• No apparent instability hazard	• No apparent damage

2.3. Observed Post-Earthquake Damage on RC Columns

In order to ensure that the method presented in this dissertation is adequate for the proposed application field of post-earthquake structural and safety assessments, it is necessary to first discuss the types of damage commonly observed and considered in the existing procedures. According to Table 2.3, distinction of damage state (RED, YELLOW, GREEN) is dependent on the following visual characteristics for RC columns: out of plumb-ness, buckling, cracking, yielding and hazard of falling. In addition, massive spalling, exposure of vertical reinforcement and large diagonal cracks extending through the column are all means for deeming a structure UNSAFE (ATC, 1989). As stated previously, these guidelines are meant to be carried out by a certified inspector or structural engineer and thus, the specifics regarding what causes a column to fall into each of these categories is a matter of judgment by the inspector based on the visible damage.

The primary concern in evaluation of seismically damaged RC columns is the loss of vertical load capacity. The loss of vertical load capacity can be portrayed in a number of different manners when the columns are part of a moment-frame system (which is the focus of this research). Hinging may occur at the top and bottom of the column (Figure 2.3), resulting in both loss of flexural strength and a significant loss in the vertical load capacity. Shear failures also occur—most often in large columns—resulting in loss of vertical load capacity (Figure 2.4). The “soft story” effect is also seen frequently in seismically damage RC frame structures where an entire story of short columns ends up failing due to the reduced effective length and thus, increased moments at either end (Figure 2.5).

Figure 2.3: Examples of top and bottom hinging in RC columns

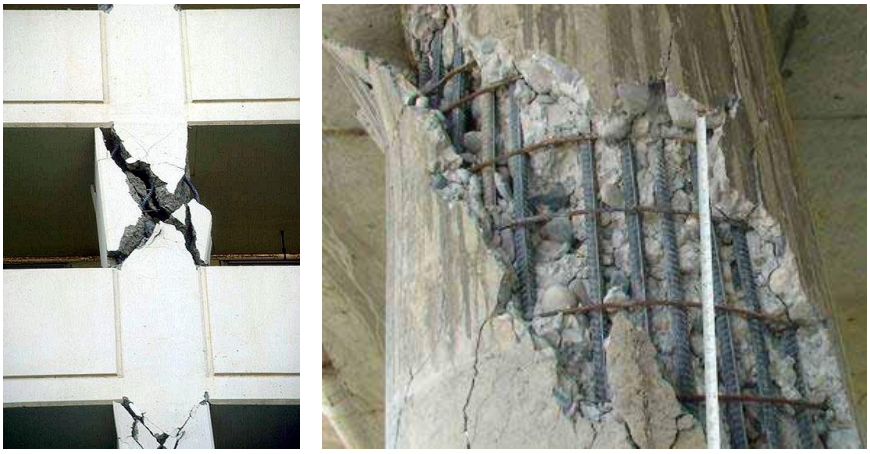


Figure 2.4: Examples of shear failure in RC columns

Figure 2.5: Examples of soft story failure in RC frame buildings

CHAPTER 3

STATE OF KNOWLEDGE

Prompted by the critical role of post-earthquake inspections, for both emergency response and occupancy, and the need for its fast performance in earthquake damaged areas, several efforts towards facilitating speedier condition assessments have been proposed. These efforts include the creation of evaluation methods based in Structural Health Monitoring, Remote Sensing and Computer Vision. These methods attempt to complement current manual practices by providing a non-subjective and quantitative assessment of the existing structural integrity of critical infrastructure and/or buildings.

3.1. Structural Health Monitoring

Structural Health Monitoring (SHM) applications in building condition assessment focus on detecting changes in the global vibration characteristics of a structure in order to identify the underlying damage (Lynch, 2007). The methods based on sensor networks use data (e.g., acceleration, deformation and strain histories) from multiple sensors installed throughout the structure prior to the earthquake to evaluate the building's condition (Kottapalli et al., 2003). Structural monitoring systems consist of sensors installed throughout a structure, retrieving characteristic measurements from each location which are then communicated to a centralized data repository where the processing occurs. The communication can be performed by way of coaxial cables or wirelessly.

Methods in SHM can be global or local. Global health monitoring methods are those which determine whether or not damage exists throughout a structure, while local health monitoring methods serve to determine the exact location and extent of the damage in the structure. Most global SHM methods focus on changes in resonant frequencies or modal shapes (Chang et al., 2003). However, some forms of damage such as the loss of a bolt in a connection, may not alter the frequency. Thus, methods can be improved by using the curvature of the mode shape, or strain, which is more sensitive to loss in stiffness due to damage in individual members (Pandey et al., 1991). The disadvantage to this suite of methods is that when the damage is distributed throughout the structure, it is difficult to locate damage when a baseline set of measurement data is unavailable. In this case, methods which focus on detecting variances in the deflection profile are advantageous (Toksoy and Aktan, 1994; Zhang and Aktan, 1995). In addition to the methods already discussed, an additional class of global SHM methods is those involving matrix updating. These methods are based on the adjustment of the mass, stiffness and damping matrices of the structure to match the measured data with minimal error. Due to the nature of these types of methods, the results are only as accurate as the baseline stiffness, mass and damping matrices, which, very often, are highly inaccurate. In addition, there are several issues with the optimization procedures since they are not unique and may not result in a positive definite stiffness matrix (Chang et al., 2003).

With regard to wired systems, the primary issues are the high cost and labor-intensive nature associated with installation. In 2001, of the 22,000 bridges in California, only 61 had been instrumented. However, the average cost per bridge for

a 60-channel system was reported to be well over $300,000 (Hipley, 2001). In addition, in the instance of an extreme weather event such as an earthquake, the connection wires, although most often protected by conduits, are highly susceptible to fracture, deeming the entire system less reliable (Kottapalli et al., 2003). Hence, due to the high system cost and labor-intensive nature of cable installation for sensor networks, wireless networks have become increasingly more prevalent (Lynch, 2007). Smart sensors, which can locally process measured data and then transmit only the necessary information are an additional potential advantage in wireless sensing systems (Nagayama and Spencer, 2007). With wireless sensor networks, the data acquisition process is more complex due to the limit in range of sensors and synchronization of the groups of sensors as a whole (Kottapalli et al., 2003). The chief drawback of wireless systems in general is that concerning power, since the remote sensor units can no longer be powered via the cables connected to the DAQ module. Instead, each remote sensor unit must also have an independent, built-in power source. Accordingly, storage and memory space are limited, and communication and processor speed is slower (Nagayama and Spencer, 2007). Thus, smart sensors are typically incapable of working in real-time.

Overall, sensor networks are installed in a very small percentage of existing structures in earthquake prone areas and rarely in the most susceptible infrastructures such as older RC frame buildings. Thus, it is evident that despite the fact that structural sensor data can reflect the state of a building, the slow adoption and associated cost have significantly hindered their practical application for building condition evaluation.

3.2. Remote Sensing

Remote sensing (RS), in general, refers to the use of aerial sensor technologies to detect and classify objects (i.e. damage) on Earth by way of transmitted signals in various forms. In general, there are two types of sensors: optical and microwave sensors. Optical sensors perceive visible lights and infrared rays (thermal, intermediate and near infrared). Further, optical sensors are associated with two methods of observation: thermal infrared remote sensing and visible/near infrared remote sensing. Thermal infrared remote sensing involves the acquisition of thermal infrared rays which are radiated from the lands surface when heated by the sun. Thus, this type of sensing is useful for observing high temperature areas (e.g., volcanoes, fires). Visible/near infrared remote sensing involves the acquisition of visible light and near infrared rays of sunlight which are reflected by various objects on the ground. With this type of sensing, information can be retrieved regarding land surface conditions (e.g., plant distributions, forests, lakes, rivers, fields, urban areas). Microwave sensors transmit and receive microwave rays which have a longer wavelength than infrared rays and visible light. The observation from this type of sensor is independent of both time and weather. Further, microwave sensors also are associated with two methods of observation: passive and active. Passive methods involve the observation of microwaves which are naturally radiated from the land surface (Figure 3.1). Active methods involve the discharge of microwaves from a sensor on the earth observation satellite followed by the observation of microwaves reflected by the land surface (Figure 3.2).

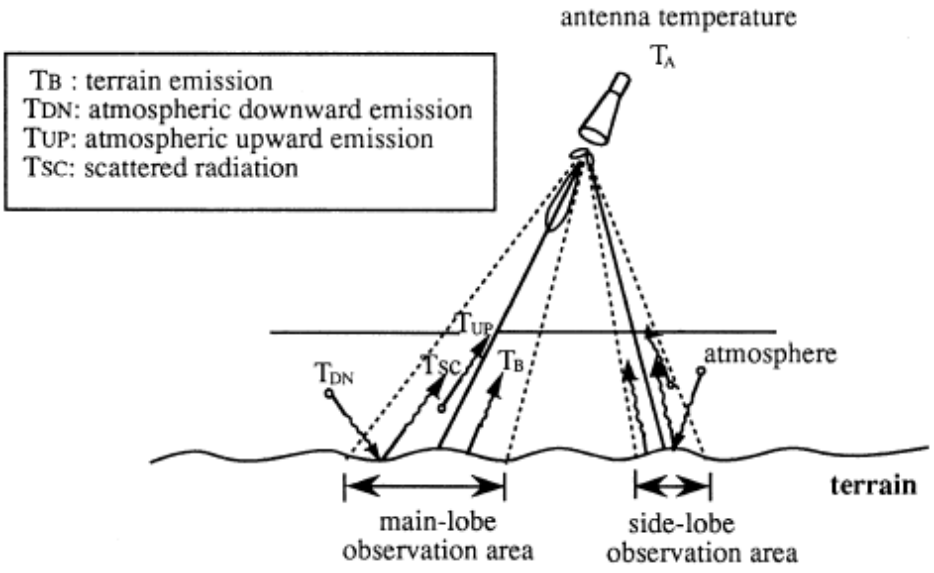


Figure 3.1: Principle of a passive microwave sensor (Murai, 1974)

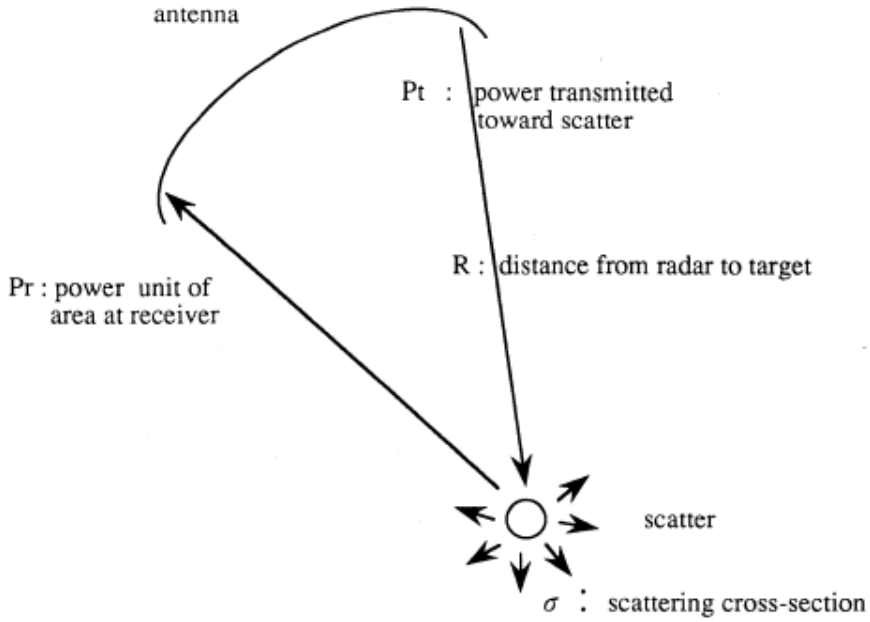


Figure 3.2: Principle of an active microwave sensor (Murai, 1974)

Several efforts have been made towards the application of remote sensing in post-earthquake building damage assessment practices using both optical and

microwave sensing methods. Prior to 2003, based on the low spatial resolution possible in satellite images, optical sensing methods were constrained to area-based damage assessments (Saito et al., 2004). Yusuf et al. (2001) developed a change detection method using pre- and post-multispectral Landsat 7 images (spatial resolution: 30m) in the Anjar area. Then, in 2002, Huyck et al. developed an automatic change detection method based on pre- and post-earthquake SPOT images (spatial resolution: 20m) of Gölcük, Turkey. In 2003, three commercially operated satellites capable of producing very high resolution images were introduced: IKONOS-2, EROS-A1 and QuickBird-2. With the capability to retrieve higher resolution images, remote sensing could be used to produce damage assessments at the individual building level (Saito et al., 2004). Using the IKONOS satellite (spatial resolution: 1m), Saito et al. (2004) identified areas of heavy damage and individual collapsed buildings after the Gujarat earthquake on January 26, 2001. Adams et al. (2005) developed the VIEWS™ (Visualizing Impacts of Earthquakes with Satellites) reconnaissance system after the Bam earthquake on December 26, 2003. Their system employed the QuickBird high resolution satellite in combination with real-time GPS readings, geo-referenced digital images and other geospatial datasets such as damage maps and street networks to help direct rescue and assessment team members to the hardest hit areas.

In addition, there have been several advancements in the application of remote sensing in post-earthquake building damage assessment practices using microwave sensing methods. Damage is detected in the remotely sensed images either indirectly or directly. Indirect methodological approaches to damage

detection infer the damage from surrogate measures such as night-time lighting levels. Hashitera et al. (1999) used satellite images from DMSP-OLS (Defense Meteorological Satellite Program-Operational Linescan System) to determine an early estimation of the area of Turkey impacted by the Marmara earthquake of August 17, 1999. Kohiyama et al. (2004) used night-time images from DMSP-OLS and employed two estimation methods (the bi-temporal images method and the time-series images method) based on the assumption that the brightness of city lights varies as a normal random variable. The method classified areas of damage far from the earthquake epicenter well.

Direct methodological approaches are based on the distinctive signature of the damage within the image. Direct approaches are either mono-temporal or multi-temporal. The mono-temporal technique is based on images retrieved after the disaster has occurred (Chiroiu and Andre, 2001; Mitomi et al., 2002), whereas the multi-temporal technique is based on spectral changes between images retrieved at various time intervals (before and after the event) (Adams, 2004). Ogawa and Yamazaki (2000) used mono-temporal, stereoscopic and single photo-interpretation of aerial photographs (taken at a 90° angle to the ground surface) to extract information regarding building damage after the 1995 Kobe earthquake and found that the method was successful at identifying severely damaged and collapsed wooden buildings. The method was considered effective at identifying the overall damage distribution but limited in recognition of damage to side walls and columns or other structural members. Hasegawa et al. (2000) also employed a mono-temporal technique. They examined aerial images captured by HDTV (high-

definition television) cameras to inspect the state of wooden and non-wooden buildings after the 1995 Kobe earthquake. The images were taken at an oblique angle, and the method was found to be effective in detecting moderate to severely damaged wooden buildings and severely damaged to collapsed non-wooden buildings.

Synthetic Aperture Radar (SAR), an active microwave imaging method, was first developed in the 1950's and is most often implemented as multi-temporal. Changes in SAR intensity and phase backscattering (taking advantage of the phase information in the return signal) have been used to detect areas of damage (Matsuoka and Yamazaki, 2002; Yusuf et al., 2002). The interferometric SAR (InSAR) specifically exploits the complex coherence of the phase of the return signal to measure the potential displacement of each pixel in the images (Chini, 2009). Guo et al (2009) used airborne and spaceborne SAR data from after the 2008 Wenchuan earthquake in China to detect collapsed urban buildings. Using only post-earthquake information, they found that longer wavelength SAR data was more effective than shorter wavelength data. Wang et al. (2009) used high resolution airborne X-band SAR images to detect building damage due to the Wenchuan earthquake and determined that it was very difficult to detect any meaningful information regarding building damage without InSAR when using only post-earthquake images and data. The disadvantage of InSAR is that the data quality and elevation accuracy are not necessarily high enough to retrieve detailed assessment information (Dong and Guo, 2012). In this case, Light Detection and Ranging (LiDAR) data is useful, where a more accurate characterization of 3D building shape signatures is possible. Dong

and Guo (2012) presented a method based on LiDAR post-earthquake data and existing (pre-earthquake) geospatial data to detect major damage to buildings automatically. With this method, building damage in the form of loss of first floors can be effectively detected (not possible with SAR or optical methods). However, as with all remote sensing methods at this time, this method is unable to detect minor damage to buildings and is only proposed for the purpose of prioritizing search and rescue and evaluation procedures in the event of the disaster (Dong and Guo, 2012).

Similar to several of the conventional methods in remote sensing, Kamat and El-Tawil (2007) proposed a method based on augmented reality (AR). This method involves a change-detection algorithm, comparing images from before and after the event in order to detect the damage as the deviation from the original. The method operates by superimposing a CAD image on top of the user's view of the real world. In this way, the post-earthquake deformed shape of the building components can be detected. The evaluation is then based on the residual, post-earthquake deformation of the entire structure. However, as with sensor technology, the prerequisite of CAD models of buildings and known locations of fiducial marks for CAD image superposition hinder its applicability in rapid and high-volume inspection situations such as in the aftermath of an earthquake (Kamat and El-Tawil, 2007). Therefore, Dai et al. (2011) proposed the use of photogrammetry to complement the AR approach, providing measurement of interstory drift induced in the damaged buildings. However, this adapted method still requires the acquisition of information regarding the building structure and geometry a priori from building owners or government databases, thus, attenuating its value and applicability in

rapid post-earthquake assessment applications where this information is not often readily available.

In general remote sensing methods of post-earthquake assessment are advantageous in comparison to other methods in that they are low-risk and offer a rapid overview or big-picture evaluation of the damage in a large geographic region (Adams, 2004). However, the technology is expensive and remains developed only so far as to obtain an overview of the damage. Since remote sensing methods involve aerial information retrieval, details regarding the existing structural state of individual buildings are not yet possible by way of remote sensing technology.

3.3. Computer Vision

The application of CV to building condition assessment employs image processing techniques to automatically detect structural elements and further to automatically detect visual indicators of damage on the structural element surfaces and retrieve properties concerning the detected damage. Therefore, each of these stages in computer vision-based assessment will be discussed in the following sections.

3.3.1. Automated Structural Element Detection

In post-earthquake safety assessments of RC frame structures, columns are considered the critical member with respect to maintaining gravity and lateral load-carrying capacity (ATC, 1989); thus, as previously noted, concrete columns are currently the focus of this framework. In order to automatically detect damage to RC frame buildings, first RC columns must be detected. Currently, this information is retrieved manually. In the recent years, there have been several efforts toward the

creation of automated structural element retrieval. Structural element detection, in general, can be described in terms of object detection. As was discussed previously with regards to damage detection, object detection methods can be classified in several ways. For the sake of this research, the following general categories of methods in object/element detection are discussed: (1) color/texture-based; (2) shape-based; (3) scale/affine-invariant feature-based; and (4) shape- and color/texture-based (a combination of (1) and (2)).

Table 3.1: Description of major color models

Color Model	Description/Method	Components	Color-Space Variations
RGB	• Additive color mixing • Describes what type of light needs to be emitted in order to produce a given color	Red, Green, Blue	sRGB, Adobe RGB, ProPhoto RGB
CMYK	• Subtractive color mixing • Describes what types of inks need to be applied so that the light reflected from the substrate and through the inks produce a given color	Cyan, Magenta, Yellow, "Key" (Black)	CMYKOG, CcMmYK
HSV	• Cylindrical coordinate representations of points in an RGB model	Hue, Saturation, Value	HSL, HIS, HSB
CIE	• Mathematical integration	Combination of cone response curves, Luminance, S cone response	CIE RGB, CIE XYZ, CIELAB, CIECAM02
YUV	• Encoded • Takes human perception into account	1 Luma, 2 Chrominance	YPbPr, YCbCr. Y'UV

3.3.1.1. Color/Texture-Based Methods

Color and texture provide a detailed description of structural members in scenarios where the member color and texture are unique. Thus, they are often good indicators for structural element detection. There are five major color models: CIE, RGB, YUV, HSV and CMYK which are discussed in detail in Table 3.1. The RGB color model is amongst the most widely used in computer vision and other image processing-based applications. Texture is typically retrieved using filter banks which are arrays of band-pass filters that serve to separate the input signal (image) into multiple components—each one carrying a single frequency sub-band of the original signal (Table 3.2).

Table 3.2: Description of common filter banks

Filter Bank	Filter Types	Number Filters
Leung Malik (LM) (Leung and Malik, 2001)	• 1st and 2nd derivatives of Gaussians at 6 orientations and 3 scales • 8 Laplacian of Gaussians • 4 Gaussians	48
Schmid (S) (Schmid, 2001)	• Isotropic (rotationally invariant) "Gabor-like" filters	13
Root Filter Set (RFS) (Varma and Zisserman, 2005)	• Edge filters (anisotropic) at 6 orientations and 3 scales • Bar filters (anisotropic) at 6 orientations and 3 scales • 1 Gaussian and 1 Laplacian of Gaussian at $\sigma = 10$ (rotationally symmetric)	38

The task of element detection using color and texture cues is typically either a one- or two-class classification problem. One-class (unary) classification problems set out to distinguish one class of objects from all other possible objects. This type of classification is typically more difficult because the training set for learning contains only the objects from the desired class. In two-class (binary) classification problems, positive and negative samples containing the desired object and the background objects are considered. The disadvantage of this type of classification is that the number of possible negative samples is infinite. There has been a significant amount of effort made towards solving both unary and binary classification problems; some of the most notable of these efforts are the following: Gaussian mixture models (GMMs), decision trees, Bayesian networks, k-nearest neighbor (k-NN), boosting, support vector machines (SVMs) and artificial neural networks (ANNs).

With respect to structural element detection, Neto and Arditi (2002) created a method which employed the combination of an edge detection algorithm and neighborhood pixel color comparison operations to retrieve structural element information. This work is only semi-automated in that the range of color values for a given structural element is manually pre-defined. In addition, the method does not go beyond the detection of the existence of structural elements within the image and fails to classify the detected element with regards to the type of structural element (e.g., column, beam, wall, etc.). Brilakis et al. (2006) used material "signatures" to retrieve information regarding the materials present within the image. The signature of clustered image regions is calculated and represented by a vector containing the regions color and texture mean and standard deviation values

(intensity, normalized red, green and blue, six response values to bank filters). The material is then determined via correlation with a material signature database using a Euclidean distance transform. Following this, Brilakis and Soibleman (2008) created a method based on the dimensions of the material regions in order to recognize linear structural elements. The maximum cluster dimension (MCD) and the maximum dimension along the perpendicular axis of MCD (PMCD) of each region was calculated, and, based on the relation between these two values, the type of structural member (column or beam) was determined. The sole reliance on color and/or texture information for structural element detection problems bears a disadvantage when seeking to detect columns or beams in frame structures where the joined members would be detected as a single structural element (Figure 3.3).

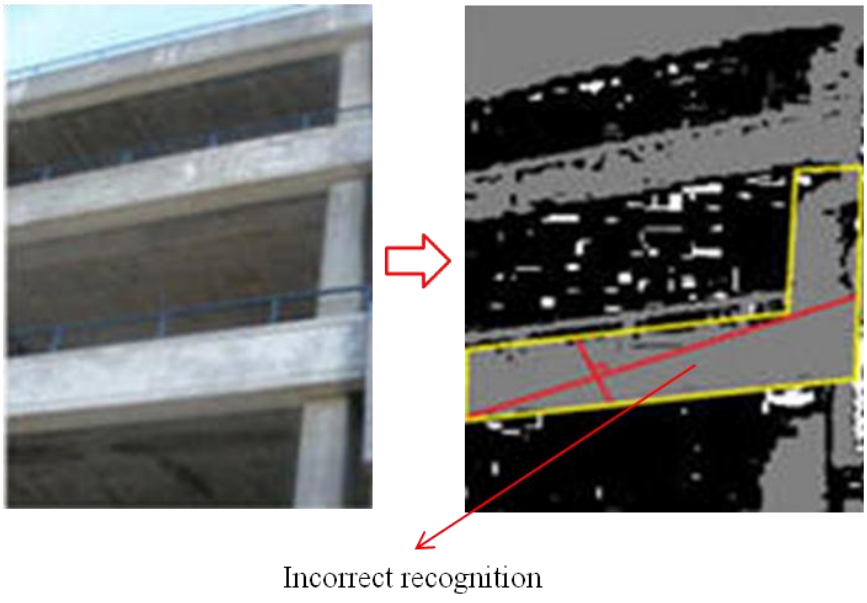


Figure 3.3: Limitation of color/texture-based element detection (Zhu and Brilakis, 2010)

3.3.1.2. Shape-Based Methods

Shape-based methods are those methods which seek to recognize objects in an image based on the object's geometry. For the most part, shape-based recognition methods begin by utilizing boundary representation via edge detection algorithms such as the Canny and Sobel operators, to describe the desired object. Edges in images are defined as the areas having sharp changes in intensity. The Canny operator is typically preferred due to its optimal design and good detection results (Guo et al., 2009). The result of an edge detection procedure is the isolated edge points rather than a continuous line of points. Thus, the following step in most shape-based methods is to employ a line detection algorithm. The Hough transform is the most widely used line detection algorithm in most applications; however, the brute force voting procedure associated with the Hough transform causes the computation time to be very costly. Thus, there have been several research efforts towards improving the efficiency of the Hough transform such as the introduction of a mean shift-based clustering stage in place of the global maxima detection portion of the existing procedure (Bandera et al., 2006) and an elliptical Gaussian kernel-based voting scheme rather than the standard brute force voting scheme (Fernandes and Oliveira, 2008). In addition to these variations of the Hough transform, there have been various other attempts towards line extraction for the purpose of element recognition—the most notable of these efforts are the following: line extraction based on the covariance matrices of the edge points (Guru et al., 2004); and, line extraction based on the principal component analysis of the

distribution of the edge points (Lee et al., 2006). However, these methods both fail in the presence of excessive noise in the image.

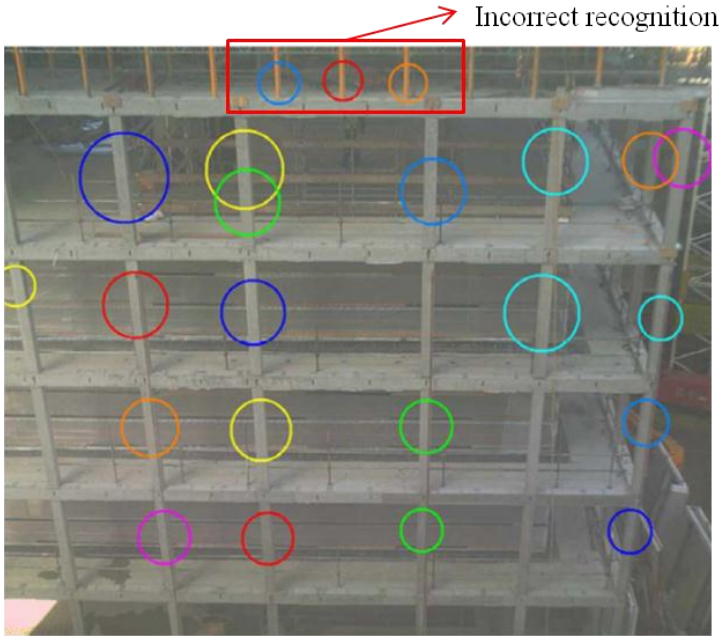


Figure 3.4: Limitation of shape-based element detection: steel railing incorrectly detected as concrete columns (Lukins and Trucco, 2007)

Regarding, element detection specifically, the results of the aforementioned line detection procedures can be used to detect 2D rectangular elements. However, elements which are rectangular in the real-world scene will appear as quadrilaterals when projected to the image scene. Thus, it is necessary for the vanishing points of retrieved lines to be estimated. Common techniques for vanishing point estimation are line clustering (Rother, 2000), Gaussian sphere (Cantoni et al., 2001) and linear least square minimization (Kosecka and Zhang, 2002). Following this stage, the quadrilaterals can be detected by way of exhaustive line pair matching (Shaw and Barnes, 2006) or graph-based search (Micusik et al., 2008). The main concerns with

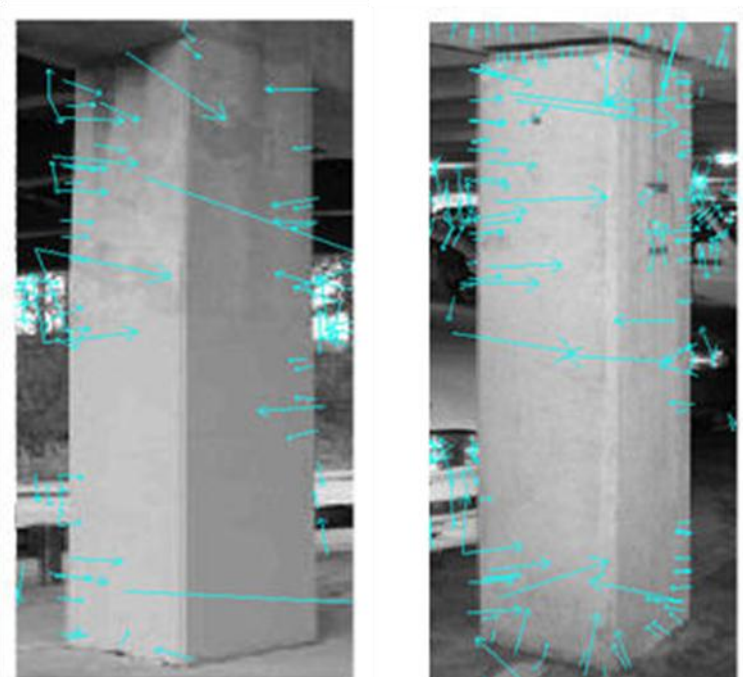
this type of procedure are that vanishing points are not guaranteed to be present in each image or video frame, the methods are time-consuming, they are only able to detect quadrilaterals which have sides aligned with the dominant scene directions and false detection rates are high since only edge information is considered (Figure 3.4).

3.3.1.3. Scale/Affine Invariant Feature-Based Methods

Scale/affine invariant feature-based methods consider a set of image features that describe the characteristics of the object of the detection in images or videos. In addition, they are invariant to scale, rotation and, oftentimes, changes in lighting conditions (Cornelis and Van-Gool, 2008). These methods can be divided into two stages: (1) feature extraction; and (2) feature matching.

In the feature extraction stage, the purpose is to provide a characterization of the element for use in the detection in images or video frames. Lowe (2004) developed the scale invariant feature transform (SIFT) which represents a local region of the image using a 3D histogram of gradient locations and orientations. In addition, there have been several other efforts and adaptations to the SIFT algorithm which are currently used for element feature extraction such as rotation invariant generalization of SIFT (RIFT) (Lazebnik et al., 2004), principal component analysis of SIFT (PCA-SIFT) (Ke and Sukthankar, 2004) and gradient location orientation histogram (GLOH) (Mikolajczyk and Schmid, 2005). In the feature matching stage, the purpose is to establish whether or not the feature vectors extracted from the images or video frames contain those extracted from the element template. This is characterized as a nearest neighbor search problem which has

been addressed using multiple randomized k-d trees (Silpa-Anan and Hartley, 2008), a spill tree (Liu et al., 2004) and a hierarch k-means tree (Nister and Stewenius, 2006). Although scale/affine invariant feature-based methods are successful at detecting a specific element in multiple images or video frames, this success does not translate to situations where multiple variations of the same element are desired for detection since no simple scale/affine transformation can accurately characterize the group any longer (Figure 3.5).



(a) (b)

Figure 3.5: Limitation of scale/affine invariant feature-based element detection: SIFT feature vectors vary for two near-identical columns (a) and (b) (Zhu and Brilakis, 2010)

3.3.1.4. Combined Color/Texture- and Shape-Based Methods

As the previous discussion clearly portrays, color/texture-based, shape-based and scale/affine invariant feature-based methods in element detection all fail to effectively detect structural element (i.e., RC columns) in images or video frames as a stand-alone method. Zhu and Brilakis (2010) proposed a method which employs elements of both color/texture- and shape-based detection for RC columns, combining column boundary information with the column color and texture cues. The method first retrieves the edge map of the image or video frame using the Canny operator and then retrieves all long near-vertical lines from the edge map using the Hough transform. Then, each of the retrieved lines is compared to the neighboring lines, and when the lengths of two neighboring lines are similar (within a threshold amount), the lines are considered a pair. The final stage of the shape-based stage of this method is the formation of a bounding rectangle for each line pair. At this point, the aspect ratios of the rectangles are calculated in order to limit the detected regions to those which have a similar length-to-width ratio of columns. Finally, the color and texture information for the regions within each proposed rectangle region are analyzed and those with properties similar to concrete are kept. The results of this method (Figure 3.6) were compared to manual detection results, and the average precision and recall were calculated to be 84.4% and 74.5%, respectively. Therefore, the detection method developed by Zhu and Brilakis (2010) for RC columns will be employed in the work presented in this dissertation in order to retrieve damage properties which are related to the size of the structural

element on which they appear and provide meaningful, quantitative damage information for the structural element at hand.

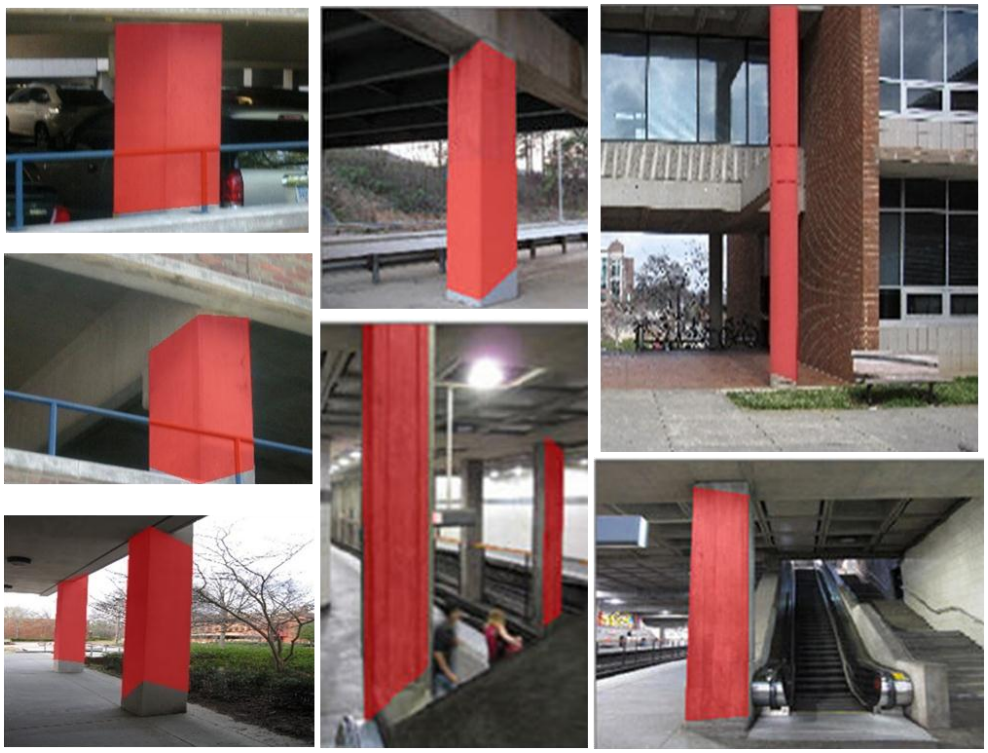


Figure 3.6: Concrete column detection results using a color/texture/shape-based detection method (Zhu and Brilakis, 2010)

3.3.2. Automated Damage Detection

Previous research in damage detection in images and/or video is substantial and comprehensive in many regards. Automated damage detection methods have been created using image processing techniques such as wavelet transforms, Fourier transforms, edge detection and region-based detection (Abdel-Qader et al., 2003; 2006; Sinha and Fieguth, 2006; Yu et al., 2007). This category of detection can be further classified into two sub-categories: (1) those which detect the presence of

damage; and (2) those which also locate the pixel area of the damage in the image, producing a "damage map."

The work already performed in crack detection as well as that in the detection of other types of damage (e.g., corrosion/rust) will be discussed in this section. At this time, there have been no known efforts in the detection of spalled regions; however, the work in corrosion detection proves noteworthy considering the visual similarities between the two damage types.

Many machine vision-based methods have been created to automatically detect the presence of cracks on concrete and asphalt surfaces. These methods distinguish the difference between cracked and uncracked regions of concrete surfaces. Liu et al. (2002) developed a crack classification system for cracking in tunnels. The method utilizes a Support Vector Machine (SVM) algorithm in conjunction with an analysis of the intensity features of cracks to differentiate regions in local sub-images as "crack," "non-crack" and "intermediate" regions. The accuracy and efficiency of the method relies on a trainable classifier based on the image center of gravity and geometry which was developed by the authors. In 2006, Abdel-Qader et al. presented a method for recognizing crack presence in bridge deck images which exercises a local principal component analysis (PCA) algorithm. Each bridge deck surface image is segmented into 16 x 16 sub-images. Each sub-image was filtered individually by linear feature detectors (horizontal, vertical and oblique) and then projected onto dominant eigenvectors. The dominant eigenvectors were pre-generated from a training data set of ten cracked and non-cracked images. The projection result was then compared with the projection

results of the training data to identify whether or not each sub-image contains a crack. This way, cracks in an image can be recognized sequentially on the basis of these 16 x 16 sub-images.

Beyond the detection of crack presence, many machine vision-based methods have also been created to automatically determine the location of the crack points within the image ("crack map"). Cheng et al. (2003) proposed a sample space reduction and interpolation thresholding approach for detecting cracks in pavement images. The thresholds were determined on the basis of a relation to the mean and standard deviation of the pixel intensities of the gray-level pavement images. However, this method contains only global processing techniques; therefore, essential crack characteristics, such as crack connectivity, are not considered. As a result, the method's detection accuracy is extensively affected by the presence of noise in the image (Yamaguchi and Hashimoto, 2010). Iyer and Sinha (2006) proposed a method for detecting and extracting cracks in contrast-enhanced sewer pipeline images. The method involves the application of both mathematical morphology and curvature evaluation techniques in order to effectively segment the image. The resulting image yields a binary crack map which could be used to measure the crack properties and determine pipe criticality levels. Sinha and Fieguth (2006) introduced a two-step detection process, including both a local and a global stage in the detection of crack pieces in buried concrete pipes. First, two crack detectors that consider relative statistical properties of adjacent image regions are applied locally to the image. These two detectors are applied in four directions (0°, 45°, 90°, 135°), and the results are then fused together. Then, a

hierarchal clustering linking and cleaning algorithm is used to connect the detected crack pieces. Yamaguchi and Hashimoto (2010) presented a scalable local percolation-based image processing method which considers crack connectivity among neighboring image pixels. The method includes termination- and skip-added procedures, considering circularity, which aid to improve upon the computational efficiency characteristic of those methods already mentioned. Their test results indicated that the method can correctly detect cracks with efficient computation time even for a large-size concrete surface image.

In addition to methods in the detection of concrete cracking, much work has been performed in the area of the detection of corrosive, or rusted, areas on steel surfaces. In general, rust detection is necessary for the application of periodic steel bridge and steel pipe condition assessments. In these practices, the existence of rusted or corroded surfaces should be determined (ASTM, 2008), and machine-vision methods exist which attempt to do this automatically. A myriad of methods use gray-scale images to detect the existence of rust defects (NFRA (Chen and Chang, 2006); BE-ANFIS (Chen et al., 2009)). In order to work in the gray-scale, the color image is converted for processing. In this process, information is lost, resulting in miscalculation of uneven background spots as rust (Chen et al., 2012). In addition, the distinct color of corroded areas with reference to the background is a defining attribute and should be considered in rust/corrosion detection. Lee et al. (2006) developed a method in rust detection which determines whether or not rust defects exist in a given image based on statistical data acquisition and multivariate analysis processing of the image in color. Lee also proposed a method based on the

comparison between eigenvalues of defective and non-defective images for the purpose of detection the presence of rust in an image (Lee, 2010).

3.3.3. Automated Property Retrieval

Beyond the detection of the existence of damage and its location in the image/video (damage map), little work has been done to automatically retrieve any further information in order to more conclusively interpret the meaning of the detected damage. The following sections discuss those efforts which have been successful in retrieving information regarding the extent of damage (crack properties and corrosion/rust properties).

There have been efforts in the field of machine-vision to retrieve information regarding the length, thickness and orientation of cracks. Chae et al. (2003) relied on an artificial neural network to retrieve crack properties in sewer pipeline images using a well-developed digital scanner—Sewer Scanner and Evaluation Technology (SSET), but it is unclear how to form the network's input data set. The effectiveness of the network is also indeterminate. Yu et al. (2007) proposed a crack detection method in conjunction with a mobile robot system for automated inspection of concrete cracks in tunnels. The method calculated the length, thickness and orientation of concrete cracks through a graph search; however, their method required the start and end points of the crack to be manually provided. In addition, their method required the robot to maintain a constant distance from the wall in order to achieve accurate measurements of the damage properties.

For the purpose of the steel bridge and steel pipe condition assessments, the percentage of the surface which is corroded is desired (ASTM, 2008). However,

similar to the case of crack property retrieval, little advancement has been made in the retrieval of corrosion properties. Chen et al. (2012) proposed a combination Fourier Transform and SVM-based method (SVMRA) which is capable of handling non-uniform illumination. They analyzed 18 color spaces, choosing the L*a*b* color space for minimal recognition error, and their method can effectively recognize rust in various paint colors and uneven illumination schemes. By dealing with the color-space rather than the gray-scale-space for the rust/corrosion images, the error pertaining to non-uniform illumination was diminished and the original image information is preserved, allowing for more accurate results. In addition the percentage of rust pixels is calculated in a classification stage. However, the processing time of SVMRA is not as fast as some methods in gray-scale detection, yielding the application of real-time assessment less probable (Chen et al., 2012).

3.4. Seismic Performance of RC Columns

In order to determine the post-earthquake vulnerability of the structure to collapse, the vulnerability of the individual RC frame members and the overall system should be considered. The seismic performance of RC columns has been studied thoroughly. However, in particular, for this research, the connection between the visible damage observed at the residual state of the structure (post-earthquake) and the performance of an RC column is of utmost significance. The remainder of this section will present two critical experimental testing programs (Sezen, 2002; Bae, 2005) as well as recent research efforts towards correlating visual damage observed throughout the loading progression with the maximum column capacity(Bearman, 2012).

3.4.1. Relevant Experimental Test Programs

Sezen (2002) studied lightly reinforced concrete columns with respect to shear and axial failure. Four rectangular columns were studied, each with a different axial load subjected: (1) 15% of the nominal strength; (2) 60% of the nominal strength; (3) varying axial load throughout the test; and (4) 15% of the nominal strength with monotonic lateral loading. The columns were loaded in an incremental fashion, and the damage was photographed and observed at each integral displacement. Bae (2005) tested five full scale RC columns in an attempt to study the impact of span-to-depth ratio and axial load on ductility and drift capacity. In these efforts, the sequence of damage to seismically loaded RC columns is well-defined, and using the information from these tests, it is possible to understand the visual damage indicators which are correlated with each stage in the progression of damage in RC columns (Bearman, 2012). In most cases, the degraded strength and stiffness of the column member throughout the loading are associated with the maximum drift (lateral load vs. displacement histories). Therefore, the vulnerability of the RC frame member can be estimated as a function of the maximum drift capacity and the response mechanism. However, in the work proposed in this research, the vulnerability of the member should be a function only of the visible damage to the member. In order to automatically estimate the drift capacity based only on the visible damage indicators—with no knowledge of the progression of damage or response mechanism, damage states stemming from these damage progressions will not be sufficient.

3.4.2. Linking Maximum Damage with Performance for RC Columns

The results of these previous studies (Sezen, 2002; Bae, 2005; Bearman, 2012) provide a basis for establishing links between damage patterns and response mechanisms for RC frame members. Bearman (2012) studied this link, providing a starting point for the research discussed in later sections for this dissertation. In the following sections, the damage progressions observed are discussed in detail.

3.4.2.1. Loss of Load-Carrying Capacity

For the purpose of this research, it is crucial that the extent of damage to the column where the load carrying capacity has been diminished is considered. The loss of axial load-carrying capacity for individual columns is the primary concern in post-earthquake safety evaluations (ATC, 1989). The loss of lateral load-carrying capacity in addition to the loss of axial load-carrying capacity should be considered in the case of columns for these evaluations. Currently, the loss of lateral load-carrying capacity is defined quantitatively, corresponding to the moment when the applied column shear drops below 80% of the maximum applied shear force (Sezen and Moehle, 2004). The loss of axial load-carrying capacity is defined as the lateral demand at which the column can no longer carry the design dead load. However, the visual cues for these metrics are indistinct. In the remainder of this section, the loss of lateral and axial load-carrying capacity are considered within the scope of the column tests, thus, associating significant indications of column failure with visual damage cues.

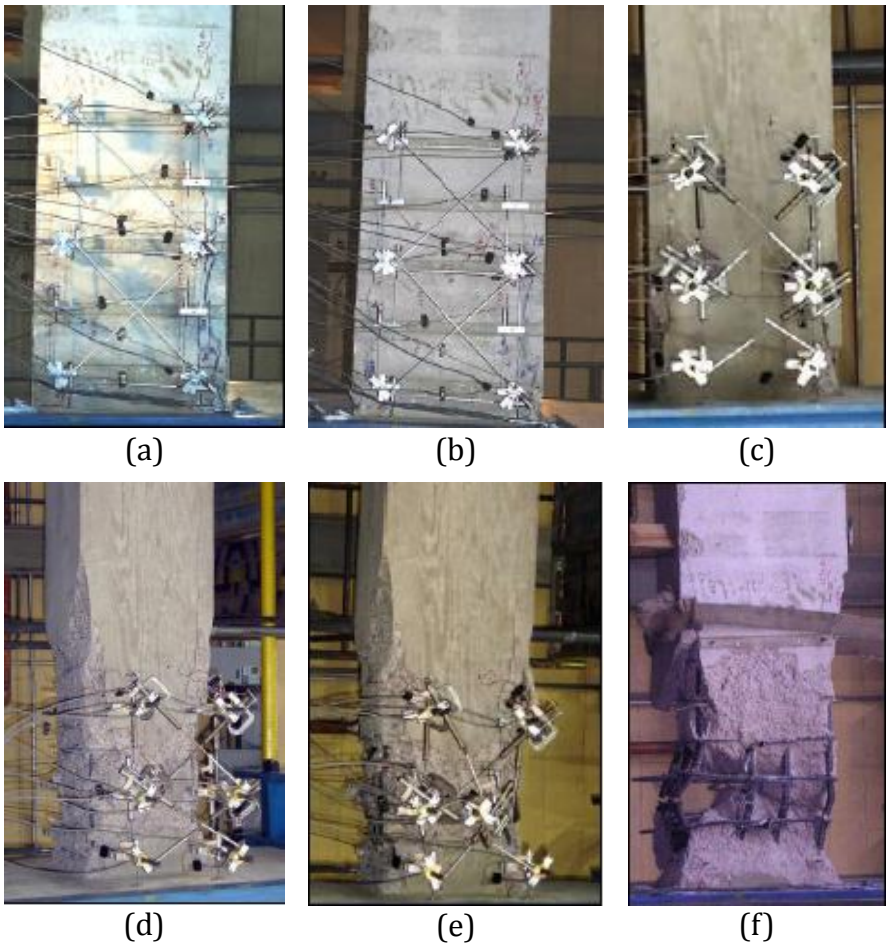


Figure 3.7: Typical flexure-critical damage progression corresponding to specific damage state indices: (a) F2/F3; (b) F4; (c) F5; (d) F6; (e) F7; and (f) F8 (Bae, 2005)

3.4.2.2. Flexure-Critical Columns

The first response mechanism studied in this work is that associated with a column which is flexure-critical. When a column has reached its full flexural capacity while the maximum shear force developed in the column is significantly below the shear strength of the column, it can be defined as a flexural failure. Bae (2005) tested five full-scale RC column specimens under seismic loads in order to investigate the relationship of the shear span-to-depth ratio, axial load level and the amount of

confining reinforcement with the column's deformation capacity. In this testing, several photos were obtained (Figure 3.7) and associated with certain points on the lateral load vs. drift history for each specimen (Figure 3.8). In combination, this information helps to postulate correlations between visible damage and the existing state of the column. In this section, the typical damage progression with respect to visually perceived damage is proposed and described for a flexure-critical RC column. The typical damage progression for a flexure-critical RC column is displayed in detail in Table 3.3.

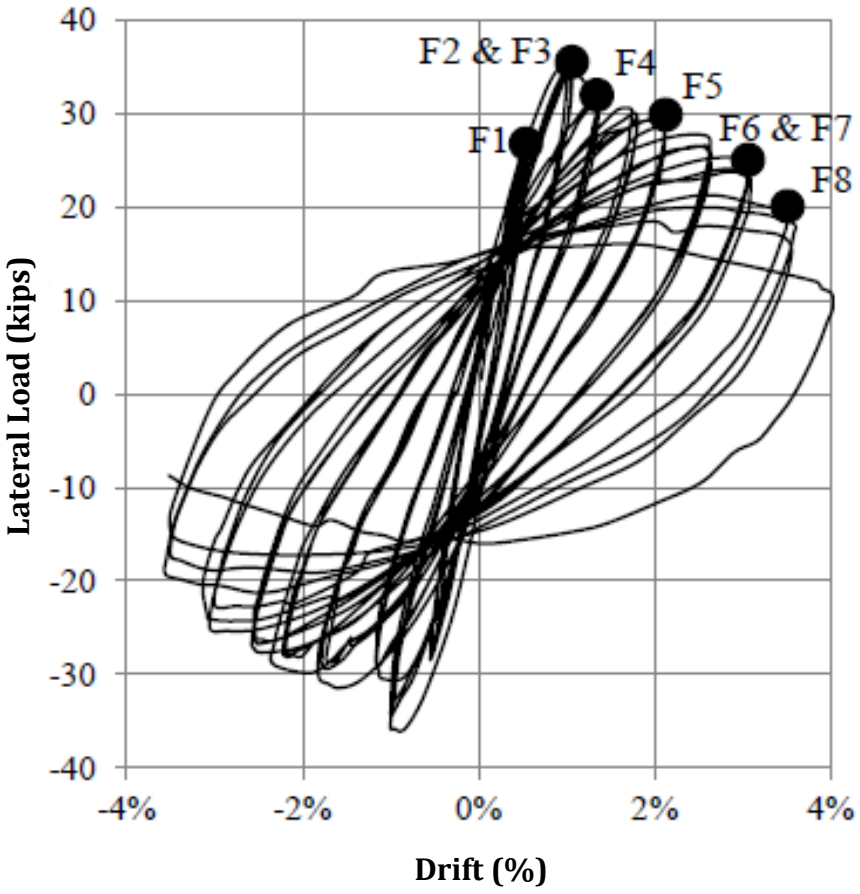


Figure 3.8: Example of lateral load vs. drift history for flexure-critical column with damage states corresponding to black points (Bae, 2005)

Table 3.3: Summary of damage progression for flexure-critical RC columns (Bearman, 2012)

Damage State	Damage Description	Drift (%)	
		LAL	HAL
F1: Flexural Cracking	• Top and bottom 1/3 of column • Perpendicular to column axis • Span width of column • Prior to spalling ≈ 0.1 in. (HAL) / ≈ 0.2 in. (LAL)	0.3	0.5
F2: Longitudinal Cracking	• Top and bottom 1/3 of column • Parallel to column axis • Prior to spalling ≈ 0.15 in.	1.0	0.75
F3: Shear Cracking	• Top and bottom 1/3 of column • At 35 to 65° from the horizontal • Initially hairline cracks (< 0.005 in.) • Prior to spalling ≈ 0.02 in. (HAL) / ≈ 0.04 in. (LAL)	1.0	0.75
F4: Concrete Spalling	• Top and bottom 1/4 of flexural faces; 1/5 of side faces • Complete spalling <= b from ends	1.5	0.75
F5: Concrete Spalling Exposing Longitudinal Steel	• Initially exposed at ≈ b/2 from ends • Exposed length ≈ b	2.0	1.0
F6: Longitudinal Bar Buckling	• Initially occurs at ≈ b/2 from ends • Total buckling length ≈ b/2	4.5	3.0
F7: Crushing of Core Concrete	• Same location as bar buckling	4.5	3.0
F8: Longitudinal Bar Fracture	• Same location as bar buckling	6.0	3.5

*where *b* = width of the column; *LAL* = low axial load; *HAL* = high axial load

The onset of flexural cracking (F1) is the first visible damage indicator in the flexural damage progression. This occurs in reinforced concrete columns when lateral loads are applied to a structure. The horizontal forces induce bending stresses, which, when in a rectangular column, cause one face to be placed in tension

and the opposite face to be placed in compression. Thus, flexural cracking will occur when the concrete tensile stress due to flexural tension exceeds the tensile strength. In appearance, flexural cracks develop perpendicular to the longitudinal axis of the column (perpendicular to the flexural tension stresses). They tend to span the width of the column on the flexural face located in the top and bottom thirds of the column. Initially, the cracks close when the column is returned to zero lateral load, but with each load cycle, the flexural cracks will continue to open and close. As the lateral load demand increases, the number and size of the cracks increases and residual cracking is observed even when the column is returned to zero lateral load. This residual crack width increases as the lateral load demand increases.

In columns exhibiting flexural response, longitudinal cracks may develop parallel to the longitudinal axis of the column (parallel to the compressive stress caused by bending of the column/in flexural compression zones) (Bae, 2005; Watson, 1989). The initiation of longitudinal cracking on the columns' flexural faces is the visual indicator for flexural damage state F2. These cracks typically initiate near the column-beam interface and propagate toward the column mid-height along the location of the longitudinal reinforcement. They remain relatively short, typically extending up to no more than a third of the column height. Longitudinal splitting cracks due to compressive loading of concrete cover may appear similar to splitting cracks associated with bond failure. Longitudinal cracks may not be observed in the field due to the cracking being closely followed by spalling at the cracks locations.

The next visible damage indicator is shear cracking on the side faces of the column (F3). Shear cracks, typically oriented at an angle of 35° to 65° from the horizontal, will propagate from the flexural faces of the column and cross the sides of the column, resulting from principal diagonal tension stresses corresponding to applied shear forces. Shear cracks are initially narrow and remain narrow under increased lateral loading for columns exhibiting flexural response.

As already mentioned in the discussion of longitudinal cracking, spalling on the flexural face occurs at the same location as the longitudinal cracking. As the lateral load increases, compressive stresses and strains increase first, causing the longitudinal cracking and then the spalling of cover concrete at the top and bottom fourth of the flexural faces. As demands increase, the spalling of the cover concrete will propagate to the side faces (parallel to the loading direction). Upon load reversal, spalling may extend over all the column faces. Concrete spalling on the flexural and side faces, or simply, spalling of the cover concrete is designated as the visual indicator for flexural damage state F4.

As the demand continues to increase, the spalling will continue such that the longitudinal steel is exposed (F5). Typically, bars on the flexural faces of the column will be exposed first, and bars on the side faces of the column may be exposed as demand increases. This exposure of longitudinal reinforcement is typically observed within the top and bottom fifth of the column.

Prior to spalling, the longitudinal reinforcement bars are surrounded by both concrete and transverse reinforcement. The longitudinal bars which have yielded in tension due to the flexural demand will buckle (F6) once the spalling of concrete has

exposed enough longitudinal bar length to reach the critical buckling load. In flexure-critical columns, bar buckling typically occurs over short lengths. Transverse reinforcement is usually spaced tightly in flexure-critical columns and effectively braces the longitudinal bars against buckling. Thus, depending on the spacing and stiffness of the transverse reinforcement, buckling may occur between ties or over several tie spaces. Visually, buckling of the longitudinal bars is marked by the curving of the bar outward from the column. In addition, lateral and axial load carrying capacity are often significantly reduced once bar buckling occurs.

As a result of the buckling of longitudinal reinforcement, the transverse reinforcement often yields, eliminating the confinement of the core concrete and allowing crushing of core concrete to occur (F7). Visually, this is defined by the spalling of concrete beyond the rebar cage. In addition, lateral load carrying capacity is often significantly reduced at the onset of core crushing. Axial load carrying capacity is lost when extensive bar buckling and crushing of the core concrete has occurred.

The final visual damage indicator is defined as fracture of the longitudinal bar(s) (F8). Occurring after longitudinal bar buckling, once the cycle is reversed and the bar is straightened again, a large tensile strain occurs on the inside of the bend. A crack may initiate during the cycles after buckling, which is then likely to propagate through the bar's cross section and fracture in subsequent cycles. Lateral load carrying capacity is significantly reduced when bar fracture occurs.

3.4.2.3. Shear-Critical Columns

Sezen (2002) tested four full-scale shear-critical, RC column specimens designed in accordance with older building codes under seismic loads. Similar to the work in flexure-critical columns by Bae (2005), in this testing, several photos were obtained (Figure 3.9) and associated with certain points on the lateral load vs. drift history of each specimen (Figure 3.10) in order to propose new models to predict the load-drift relations and shear strength of the columns tested. In combination, these photos and the experimental results from the cyclic testing, help to suggest correlations between visible damage and the existing state of the column. In this section, the typical damage progression with respect to visually perceived damage is proposed and described for a shear-critical RC column. The progression of typical damage in a shear-critical column is displayed in full in Table 3.4.

(a) (b) (c) (d)

Figure 3.9: Typical shear-critical damage progression corresponding to specific damage state indices: (a) S1/S2; (b) S3.0/S3.1; (c) S3.2; and (d) S3.3 (Sezen, 2002)

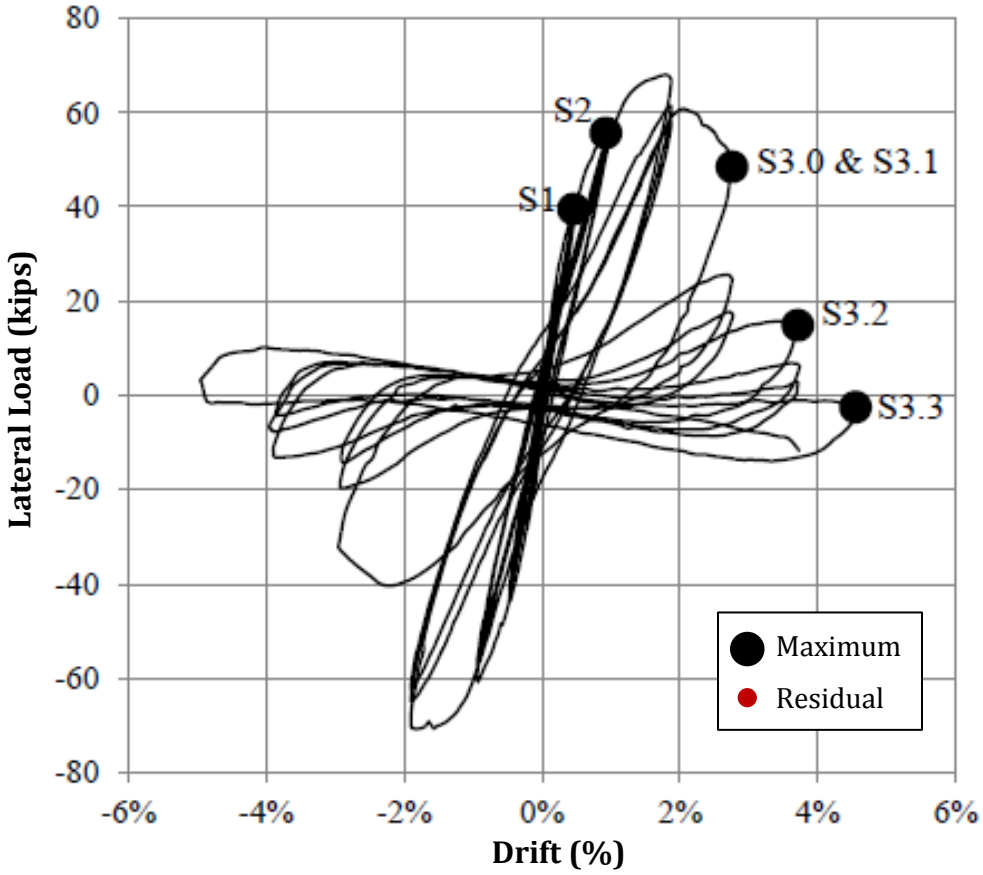


Figure 3.10: Example of lateral load vs. drift history for shear-critical column with damage states marked by black points (Sezen, 2002)

The description of the flexural and longitudinal cracks is identical to that of flexure-critical columns as presented in Section 3.4.2.2 for F1 and F2 since columns which are shear-critical respond to low lateral loads in flexure. The damage in shear-critical columns initiates with flexural and longitudinal cracking (S1). Following the propagation of flexural and longitudinal cracking, the column will experience shear cracking on the side faces (S2). The description of these shear cracks is similar to that in the flexure-critical columns. However, the difference here is that for shear-critical columns, shear cracks may form at any location along the

column's height. As the displacement demand increases, the width of the shear cracks will increase.

Table 3.4: Summary of damage progression for shear-critical RC columns (Bearman, 2012)

Damage State	Damage Description	Drift (%)	
		LAL	HAL

S1: Flexural and Longitudinal Cracking	• Flexural cracks:
	  • Same as F1 (HAL)
	  • Prior to S3 ≈ 0.05 in.
	• Longitudinal cracks:
	  • Same as F2 (HAL)
	  • Prior to S3 ≈ 0.1 in.
		0.25	0.25

S2: Shear Cracking	• Same as F3 (HAL) • May occur at any height	0.5	0.5
S3.0: Widening and Localization of Shear Cracks	• May occur at any height • At 35 to 65° from the horizontal • Initially hairline cracks (< 0.005 in.) • Prior to spalling ≈ 0.3 in. residual (HAL) / ≈ 0.5 in. residual (LAL)	2.0	1.75
S3.1: Longitudinal Cracking on Side Faces	• May run the entire length of the column • Meet localized shear cracks near edge • Prior to spalling ≈ 0.5 in. residual	2.5	1.75

S3.2: Concrete Spalling on Side Faces	• Possible spall shapes:
	  • Triangle where shear and longitudinal cracks meet
	  • Parallelogram encompassing primary shear cracks
	• Edges of spall are at 35 to 65° from the horizontal
	• May occur at any height
		2.5	1.75

S3.3: Longitudinal Bar Buckling	• May occur at any height	2.5	1.75
S3.4: Crushing of Core Concrete	• Typically occurs with bar buckling • May occur at any height	2.5	1.75

*where *b* = width of the column; *LAL* = low axial load; *HAL* = high axial load

At this point, the remainder of the damage categories described herein are associated with a brittle shear failure mechanism in the column. The experimental tests by Sezen (2002) indicate that these damage types tend to occur suddenly and approximately simultaneously once the earthquake shear demand exceeds the column capacity. A review of images from the NISEE Earthquake Engineering Online Archive (http://nisee.berkeley.edu/elibrary/) suggests that because damage states occur approximately simultaneously, only the final damage state may be observed in the field following an earthquake.

Prior to failure, shear cracking localizes with one or two wide shear cracks (S3.0). The wide shear cracks have the same orientation as the initial shear cracking, and in the field, one or two wide shear cracks may indicate shear failure of the column. In addition, some shear cracks will propagate towards the edge of the side face and change orientation such that they are vertical, becoming longitudinal cracks. This longitudinal cracking on the side faces (S3.1) aligns with the location of the longitudinal steel at the edges of the column. They are likely to run the length of the column prior to failure. If the displacement demand increases rapidly, longitudinal cracking is typically not evident and thus, if these cracks are not visible, that should not be deemed as an indication that the column is not nearing failure. However, spalling of concrete on the side faces of the columns is likely to be visible (S3.2). Regions dominated by shear and longitudinal cracks on the side faces are likely to spall. These regions will typically be triangular, bounded by the longitudinal and diagonal cracks, or parallelogram-shaped, encompassing the diagonal cracks primarily.

Laboratory testing has shown that the onset of concrete spalling occurs simultaneously with the loss of lateral load carrying capacity (Sezen, 2002). In conjunction with concrete spalling, longitudinal bar buckling will often occur (S3.3), and significant spalling and bar buckling are followed closely by the loss of lateral load capacity. Once the longitudinal bars have buckled, as is the case with flexure-critical columns, the transverse reinforcement will typically yield. This yielding of transverse reinforcement allows for crushing of the core concrete (S3.4). Once the core crushing has initiated, the axial load carrying capacity is lost.

These damage progressions (for flexure- and shear-critical RC columns) display the relationship between visible damage which can be observed on the column surface and the existing state of the column; however, the visible damage is observed at the maximum drift values. A relationship still needs to be defined between the visible damage observed at the residual state of the column and the maximum column capacity.

3.5. Closure: Gaps in Knowledge

There have been several efforts directly related to the advancement of post-earthquake assessment practices. SHM has been employed in the field of post-earthquake structural assessments. However, the cost of sensors, in addition to the cost and time associated with sensor installation has been a hindrance in the success of this approach. Remote Sensing type approaches have also been proposed, superimposing CAD drawings or high resolution, aerial images with post-earthquake images to determine the residual state of the building. These approaches, similar to SHM, are associated with high costs and prior knowledge

concerning the damaged structures. Hence, each of these types of approaches, as a stand-alone, is an unlikely solution in the field of post-earthquake evaluations where a large number of buildings need to be evaluated in a rapid manner.

In addition, much work has been performed in the application of computer vision to post-earthquake evaluations. Methods in structural element detection, damage detection and property retrieval have been created with respect to cracking on pavement, tunnel, bridge and column surfaces and corrosion on bridge surfaces. These approaches, although the bulk of those which currently exist do not yield quantitative damage information pertinent to post-earthquake building assessments, do in fact provide promise for automated, real-time and quantitative evaluations without any prior knowledge of the buildings structure, design or state. Finally, even with a comprehensive retrieval of pertinent and quantitative damage information using computer vision techniques, a valid relationship between visible (residual) damage observed on a structure after an earthquake occurs and the existing condition or capacity of that structure has yet to be established. Therefore, fast, reliable and automated evaluation of the safety and integrity of damaged buildings after earthquakes which, simultaneously, does not disrupt the remaining tasks of the inspector, has yet to have been achieved. This kind of evaluation can be made possible by reinforcing existing inspection procedures with an automated tool which can provide a cost-effective, quantitative and real-time assessment without introducing any further communication or organizational concerns. However, in achieving this goal, the following research questions should be addressed:

1) How can the current evaluation procedures be improved such that the results are no longer purely qualitative?
2) How can a building be evaluated accurately and efficiently without any prior knowledge of the buildings' specifics?
3) How can the relevant visible damage on structural surfaces be automatically detected?
4) How can the detected damage be automatically correlated with the structural element in order to retrieve the necessary properties?
5) Can a reasonably reliable estimate of the damage state of the RC column be automatically determined based only on this detected visible damage?

CHAPTER 4

DAMAGE DETECTION AND PROPERTY RETRIEVAL

Based on the research objectives and existing gaps in knowledge, the need for real-time, quantitative and efficient assessment practices can be addressed by the creation of an automated assessment tool which utilizes computer vision to immediately determine the state and safety of the structural components throughout the building. This chapter discusses the necessary steps for the automated detection and retrieval of key properties pertaining to the visible damage on RC columns within the context of a framework for automated post-earthquake building evaluation (tagging, loss estimation and collapse probability) (Figure 4.1). The proposed methodology begins with the collection of video frames via a high-resolution video camera which are then transmitted to a computer off-site for analysis. There, each frame is searched for RC columns and the damage inflicted on the columns. The spatial damage properties are considered collectively, so that the column's existing strength/stiffness can be approximated in the form of a damage index. In a global sense, at this point, the individual damage states of the columns, in combination with the building's structural type and column arrangement per floor, will be used to query a fragility database constructed from analyses of experimental data. The database contains building fragility curves that report the probability of various levels of structural damage. In consulting these curves the estimate of the probability of the entire structure to collapse in the event of an aftershock will be determined.

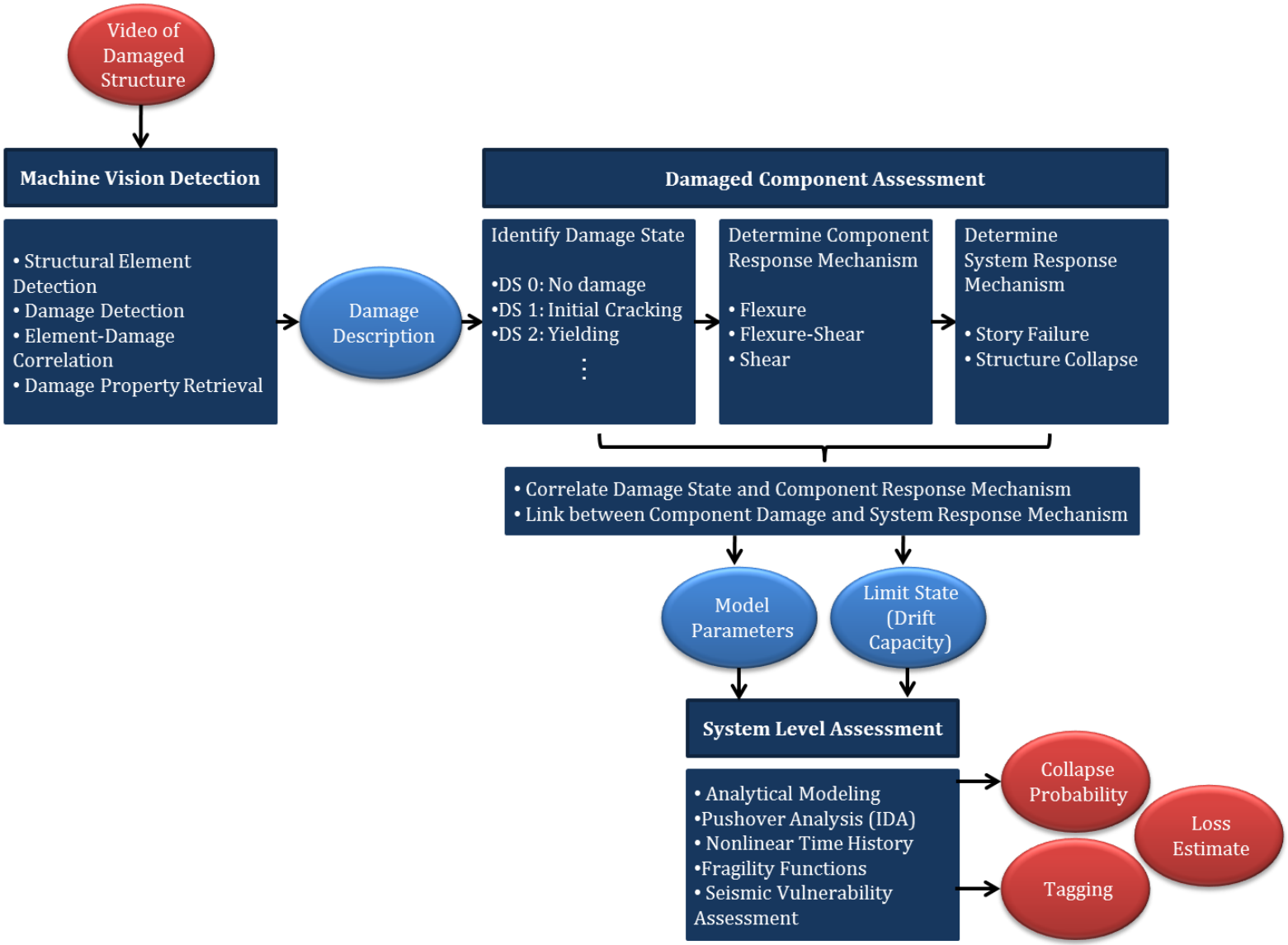


Figure 4.1: Overall framework for automated post-earthquake safety/structural evaluation

In specific, this chapter first presents the method in structural element detection which is employed in this research. Following this brief discussion, two novel methods in the area of damage detection and property retrieval are presented. The first method is focused on the detection and property retrieval of spalled regions on concrete element surfaces. In this method, first a local entropy-based threshold is applied to the image, producing the spalled map. The extent of spalling into the column is calculated primarily based on the amount and type (transverse or longitudinal) of reinforcement exposed. The regions of reinforcement are detected using a threshold in the CMYK color-space. Then, the exposed longitudinal reinforcement is detected using a state-of-the-art template matching algorithm, and the exposed transverse reinforcement is detected using a state-of-the-art region growing technique which was developed for crack detection and is based in percolation theory. In addition, a binary image thinning algorithm and distance transform are applied to the reinforcement maps in order to retrieve specific properties pertaining to individual exposed bars or segments of bars.

The second method is focused on the property retrieval of cracks and the definition of crack patterns on concrete element surfaces. Once the crack map is retrieved (Yamaguchi and Hashimoto, 2010), a binary image thinning algorithm and distance transform are applied in order to retrieve the crack skeleton and distance field for each crack segment. Following this, with the aide of object bounding boxes, crack length, width, orientation and spacing are all calculated. With this information, the dominant crack orientation can be determined, and the crack pattern specified.

4.1. Concrete Column Detection

In post-earthquake safety assessments of RC frame structures, columns are considered the critical member with respect to maintaining gravity and lateral load-carrying capacity (ATC, 1989); thus, as previously noted, concrete columns are currently the focus of this framework. Concrete columns (typically rectangular or circular in cross-section) are man-made solid objects that have two distinguishing visual characteristics: 1) their shapes are dominated by long vertical boundary edges, and 2) the uniform texture and color pattern of concrete on each of their surfaces. Based on these characteristics, the detection method (discussed in Section 3.3.1.4) developed by Zhu and Brilakis (2010) for concrete columns will be employed. It is necessary to automatically detect the RC columns on which the damage exists in order to ensure that the method in automated damage index estimation is fully-automated. The damage properties will be quantified by way of providing measurements of the damage in relation to the measurements of the column on which the damage exists. With the implementation of this automated method in concrete column detection, the methods in automated damage detection and property retrieval can be constrained to those regions of the image which are positively recognized as columns. In other words, only those columns which are correctly detected in an image or video frame will be searched for damage. This is a significant advantage in developing algorithms for damage detection and property retrieval because the potential for false positive detection results (pixels in the image or video frame which are incorrectly recognized as the desired object of the detection procedure) is minimized such that it includes only the non-damaged

pixels on the RC column surface rather than every pixel in the image in the image which is not a damaged pixel. Hence, for future reference, the "background" which is commonly referred to throughout the remainder of this document is simply the non-damaged RC column pixels.

4.2. Spalling Detection

Spalling is a significant indicator of the extent of damage to an RC structural element. Several of the current visual evaluation metrics for post-earthquake safety and structural assessment concern the degree of spalling on column surfaces, and so, it is the object of the first computer vision damage detection and property retrieval method created for the purpose of this work.

Based on the appearance of spalled regions on RC columns (Figure 4.2), the texture of the concrete pixels is the most distinctive visual characteristic. In contrast to the unspalled concrete surface pixels, those in the spalled region are coarser in nature. Entropy is a statistical measure of randomness. In image processing, it can be used to characterize the texture of an input image. In the method which is presented in this framework, the spalled map will be detected using a local entropy-based thresholding algorithm. The algorithm is local rather than global because the contextual information for each pixel is of utmost significance (the distinction between unspalled and spalled regions). Thus, the method created for the detection of spalled regions is an image segmentation algorithm based on a local entropy-based threshold. This specific algorithm will segment the image into two regions: *background* and *foreground*, or *concrete surface pixels* and *spalled pixels,* by calculating the entropy of the pixels in the 9 x 9 region surrounding the pixel of

interest. Since a higher value of entropy indicates a greater tendency towards chaos, if the entropy is above a designated threshold value, the pixel of interest is designated as *foreground/spalled pixel*; otherwise, the pixel of interest is designated as *background/concrete surface pixel.*

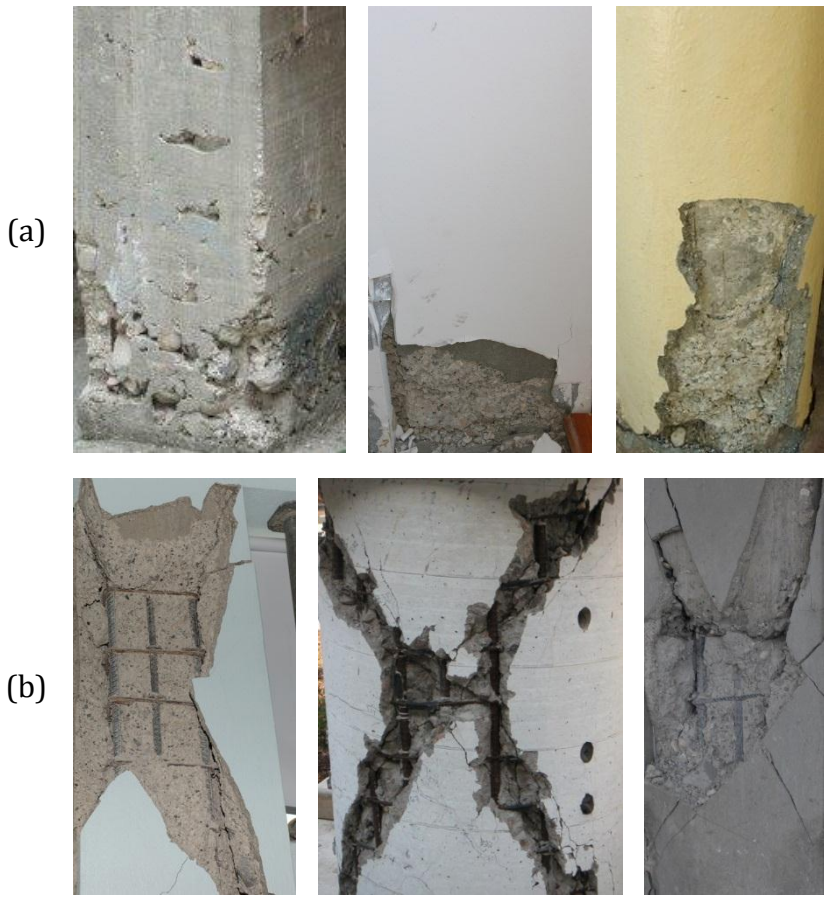


Figure 4.2: Examples of spalled concrete images: (a) spalling which only exposes the cover concrete; and (b) spalling which exposes reinforcement

This type of segmentation algorithm is deemed successful in situations where the variation in local content of the image pixels in one region (the image pixel neighborhood) is high. With careful observation of a sample space of images

(i.e., Figure 4.2), it is evident that this is in fact the case; spalled regions, whether spalled such that only cover concrete is visible (Figure 4.2 (a)) or to the extent where reinforcement is also exposed (Figure 4.2 (b)), typically display wide variations in intensity on a pixel-to-pixel (local) basis, whereas the non-damaged regions display a larger magnitude of continuity across pixel neighborhoods.

The local entropy-based thresholding method calculates the Shannon entropy value for each image pixel given a designated neighborhood (Equation 4.1) (Shannon, 1948), where N is the neighborhood dimension, $p_{i,j}$ is the probability of the intensity value of the pixel occurring at row i, column j in the neighborhood based on all of the pixel intensity values in the neighborhood. Thus, an array will be created consisting of entropy values attributed to each pixel in the input image. This array will then be thresholded against the average value plus one standard deviation in the array according to Equation 4.2, where $I_{m,n}$ is the pixel at row m, column n of the resulting image, $E_{m,n}$ is the entropy of the pixel at row m, column n and T is the average entropy in the neighborhood plus one standard deviation in the entropy of the neighborhood.

$$E_{m,n} = -\sum_{i=m-N/2}^{m+N/2} \sum_{j=n-N/2}^{n+N/2} p_{i,j} \log_2(p_{i,j}) \qquad \text{(Eq. 4.1)}$$

$$I_{m,n} = \begin{cases} 255 : E_{m,n} > T \\ 0 : E_{m,n} \leq T \end{cases} \qquad \text{(Eq. 4.2)}$$

In order to acquire a higher level of confidence at this stage in the detection of the spalled region, the result is processed by a sequence of morphological closing and opening operations. This serves as an intermediate post-processing step, de-noising the result. A rectangular (square) shaped structuring element the same size (9 x 9) as the neighborhood for the local entropy-based thresholding is employed.

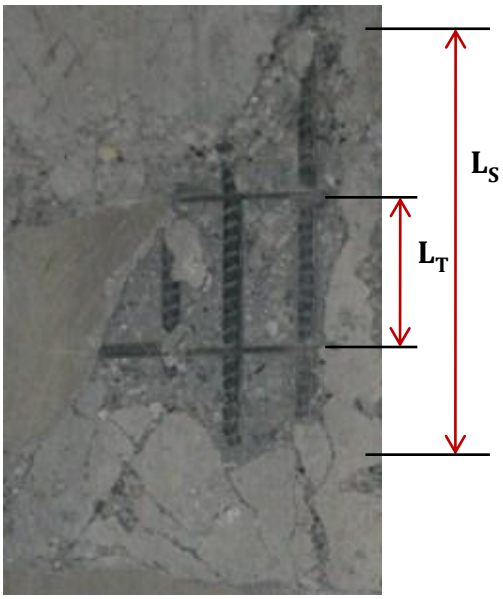


Figure 4.3: Specification for spalled length classifications

4.3. Spalling Property Retrieval

Once the spalled map is detected, both the length and the depth (into the column) of the spalled region should be evaluated in order to quantify the extent of spalling on the column. For the purpose of this work, depth of spalling will be classified with respect to the following five categories: (1) no spalling; (2) spalling of concrete cover which does not expose reinforcement; (3) spalling which exposes transverse reinforcement; (4) spalling which exposes longitudinal reinforcement; and (5)

spalling which exposes both transverse and longitudinal reinforcement. The length of spalling along the column is depicted by the relative length of the extent of spalling along the vertical (longitudinal) direction of the column L_S, as well as the length between the extreme exposed transverse reinforcement bars L_T (Figure 4.3). The overview for the method in spalling detection and property retrieval is displayed in Figure 4.4.

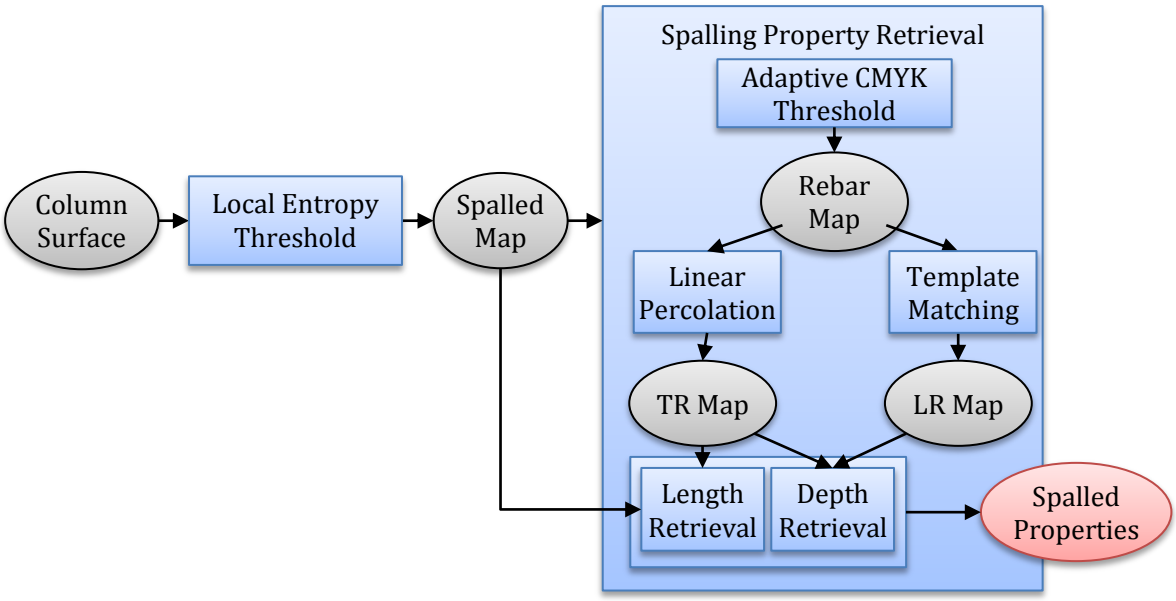


Figure 4.4: Overview of method in automated spalling detection and property retrieval

4.3.1. Reinforcement Detection

In order to effectively quantify the extent of spalling on the surface based solely on the visible condition of the element, the type and amount of reinforcement which is exposed is considered.

Considering the visual appearance of reinforcement, three main distinctive visual characteristics have been identified: (1) ribbed texture along the surface of the potentially exposed reinforcement; (2) the distinct color of steel; and (3) the width of the longitudinal/transverse reinforcing steel should be significantly less than the width/height of the column. Initially, the distinct color of steel is considered in order to retrieve the overall reinforcement map (including, but not distinguishing between, transverse and longitudinal reinforcement).

Due to the distinction in the CMYK color-space between the reinforcement pixels and the background/concrete pixels, image segmentation by way of thresholding is an appropriate means of processing the image data. After inspection of the histograms of several images, it became evident that a global threshold would be effective, adaptive to each image, and each channel (Cyan, Magenta, Yellow, Black), according to the specific mean and standard deviation values. In this method, criteria for the partitioning of the image were studied further in order to find that with the highest performance. In these studies it was found that a double threshold (Equation 4.3) where $I\ (m,n)$ is the pixel at row m, column n of the resulting image, T_1 is the mean pixel value minus two standard deviations, T_2 is the mean pixel value for each image channel and $I_{orig}\ (m,n)$ is the original pixel value in the image channel at row m, column n. With the completion of this algorithm, the texture of the reinforcement will be highly distinct within the RC element surface. The result is five binary images: one for each of the C, M, Y and K image channels and one which is the result of combining these four results with a simple *OR* operation such that if a pixel is detected in any one of the four channels, it is represented in this

combination image. The combination image is specified as the region of interest (ROI) for the remainder of the processes in transverse and longitudinal reinforcement detection (i.e., if a pixel is not detected as reinforcement in this image, it will not be detected in the later processes of longitudinal or transverse reinforcement detection).

$$I(m,n) = \begin{cases} 255, T_1 \le I_{orig}(m,n) \le T_2 \\ 0, otherwise \end{cases} \quad \text{(Eq. 4.3)}$$

4.3.1.1. Longitudinal Reinforcement Detection

Once the map representing any indication of exposed transverse and longitudinal reinforcement is produced, the types of reinforcement should be made distinct. In images and video data, due to the larger size of longitudinal reinforcing bars in RC columns, the texture is more prevalent on longitudinal reinforcement. However, the texture is not an adequate visual indicator for transverse reinforcement since the ribs are hard to recognize in medium to high resolution images. Therefore, the methods in longitudinal and transverse reinforcement detection differ.

In order to properly consider the discernible texture of longitudinal reinforcement, the method in longitudinal reinforcement detection employs the OpenCV template-matching algorithm in each of the four thresholded results (excluding the combination thresholded image) (OpenCV, 2010). Although the texture of reinforcement is a dominant visual characteristic, it can vary in appearance according to orientation and arrangement of both the ribs on the bars as

well as the rebar themselves. Thus, the resulting image, post-thresholding, will vary significantly from one image to the next. To account for this, the algorithm is applied with a diverse assortment of binary templates consisting of a small section of rebar (Figure 4.5), facilitating the collection of the response of the reinforcement texture regardless of variation in orientation or types of ribbing. The results of each template match for each channel are then hard-thresholded according to the matched image's average pixel value and standard deviation (Equation 4.4) where $I(i,j)$ is the pixel value of the image after thresholding, $I_{orig}(i,j)$ is the pixel value of the image before thresholding, μ is the average image value and σ is the standard deviation in the image values. This will result in another binary image for each channel, each accumulating the areas where the match is strongest and relieving those where the match is weakest.

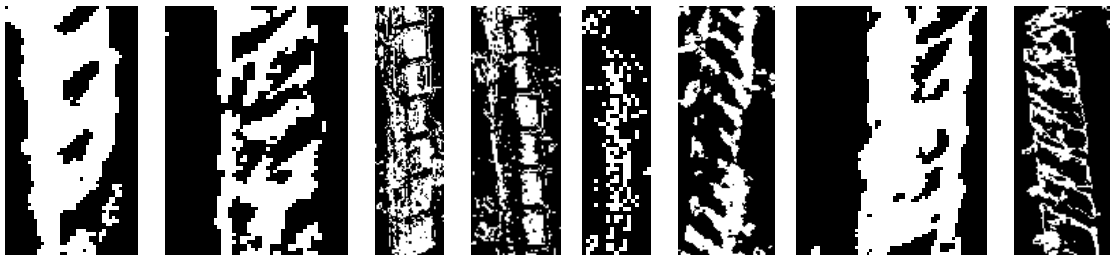


Figure 4.5: Binary longitudinal reinforcement templates for template matching algorithm

$$I(i,j) = \begin{cases} 255, & I_{orig}(i,j) \leq \mu + \sigma \\ 0, & otherwise \end{cases} \quad \text{(Eq. 4.4)}$$

Once each of the matched images is processed, all of the matched images for a single channel are combined into one resulting for each type of reinforcement using another *OR* operation. Then, the combined matched images for each channel are further combined using a simple *AND* operation such that only those pixels which are detected in each of the four channels are translated to this resulting image. Finally, the detection results are de-noised in order to provide a higher level of confidence in the detection results. The de-noising process consists of further morphological operations. In each of these operations, the size of the structuring element is determined such that it is proportional to the size of the image/detected column in order to account for the known characteristic of reinforcement concerning its relative size within an RC column. In this process, two things occur: (1) regions smaller than the reinforcement are removed; and (2) regions within highly detected areas (highly likely to contain reinforcement), which are undetected, are filled.

4.3.1.2. Transverse Reinforcement Detection

Since the texture of transverse reinforcement detection is not as distinct, the method in transverse reinforcement detection considers only the shape and color attributes of the reinforcement within the context of a concrete column. The method is an adapted version of the percolation-based method proposed by Yamaguchi and Hashimoto (2010) which was alternatively adapted for the purposes of this research for crack detection. First, the Canny operator is applied to the image to find the pixels with highest gradient magnitude. In addition, the CMYK image is blurred

using a 7x7 Gaussian blur. This softens the effect of any texture that is visible, such that the percolation process is more effective.

Once the edge points are retrieved, the percolation detection is initiated at each pixel location (designated in the edge image) in each channel of the smoothed CMYK image. Based on this method, the process is the following: (1) an edge pixel is selected to initiate the percolation process; (2) this pixel is added to the region, D_p; (3) the neighboring image pixels of those pixels in D_p are added to the region, D_c; (4) all image pixels in D_c which have image intensity (C, M, Y or K value) less than that of the original pixel (the maximum C, M, Y or K value) in D_p are added to D_p; (5) return to (3) for each pixel added in (4). This process continues until no more pixels can be found with lower image intensity (C, M, Y or K value) than those in D_p. At this point, the circularity, F_c of D_p is calculated (Equation 4.5), where C_{count} is the number of pixels in D_p and C_{max} is the maximum dimension of the rectangle bounding D_p. This measure is an indication of the shape of the detected region of pixels. As F_c approaches zero, the shape of the region tends towards linear. Conversely, as F_c approaches one, the shape of the region tends towards circular. Thus, for the detection of transverse reinforcement, regions with a circularity measure less than 0.18 and an approximate angle between -20° and +20°, all of the pixels in D_p are marked as transverse reinforcement pixels.

$$F_c = \frac{4 \times C_{count}}{\pi \times C_{\max}^2} \qquad \text{(Eq. 4.5)}$$

Once this process is completed for each channel, the results for each individual channel percolation are combined using a simple *OR* operation, such that if a pixel is detected as transverse reinforcement in any of the channel percolation procedures, it is also defined as transverse reinforcement in the combined image. At this point, each individual piece of transverse reinforcement must be isolated in order to retrieve the value, L_T. In order to do this, a binary image thinning algorithm (Cychosz, 1994) is used to retrieve the centerline of each detected region. This portion of the algorithm is commonly termed "skeletonization" since the centerlines are equivalent to skeletal-like parts of the regions. In addition, a Euclidean distance transform (Fabbri et al., 2008) is employed, calculating the distance field (or number of pixels from the center pixel to edge pixel for each skeleton point) for each detected region.

At this point, the connectivity of the detected points has to be considered in order to correctly segment the regions. This process is illustrated in Figure 4.6, and explained by the following: one transverse reinforcement skeleton point is visited, and its neighbors (surrounding pixels) are checked. If there is a point designated as a transverse reinforcement skeleton point connected to the point, the current segment grows by including that neighboring skeleton point. When all skeleton points are visited, the segments' directions, start points and end points are checked. Any two segments are merged if they have the same direction, and the end point of one segment is specified as the start point of the other. However, for transverse reinforcement, the segmentation most often differs from that displayed in Figure 4.6 with respect to the branching, or points where segments of varying orientations

intersect. For transverse reinforcement, the main purpose of the segmentation is to isolate the individual bars within the image, and remove pixels which may be false negatives (falsely detected transverse reinforcement pixels). The regions have already been filtered according to orientation such that only horizontal regions are detected; therefore, branching of reinforcement pixels is rarely encountered.

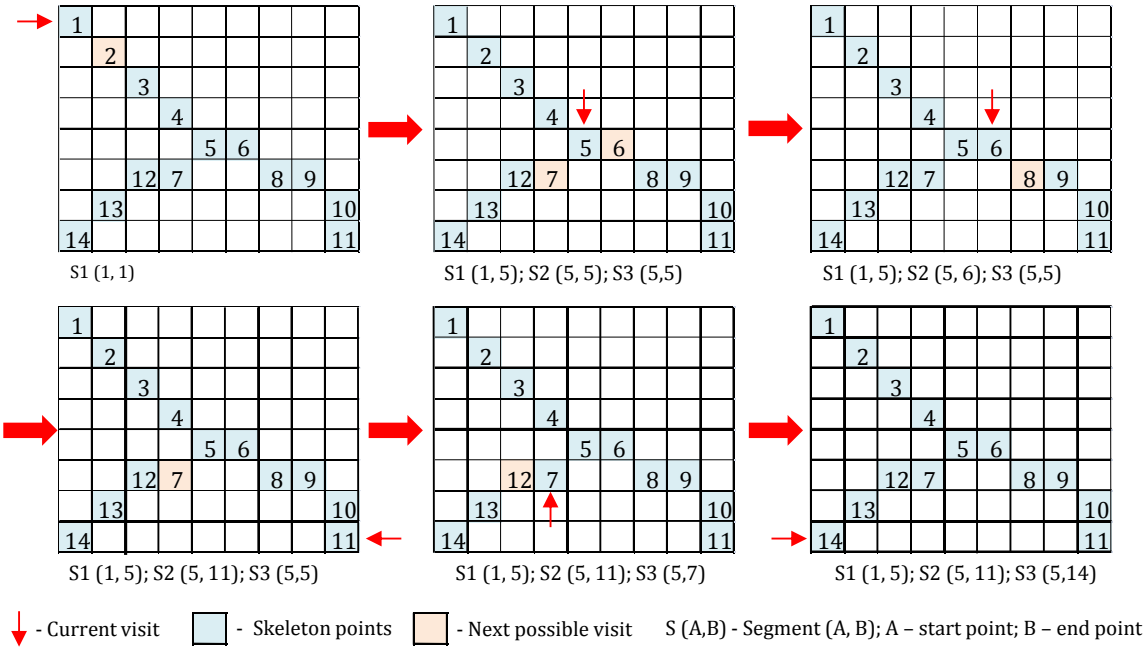


Figure 4.6: Damage skeleton segmentation

4.3.2. Quantification of Spalled Properties

Once the spalled region and any amount of exposed transverse and longitudinal reinforcement have been detected, the extent of spalling on the RC column should be quantified. The measure of how extensive the spalling is into the column (depth) is calculated by way of a classification stage. The column is dispensed into one of the five categories presented at the beginning of this section and further explained in

Table 4.1. This table also displays the measure(s) which help to classify the extent of spalling along the length of the column that can be retrieved in each category.

Table 4.1: Spalled depth classifications

Category	Description	Measure(s)
S0	No spalling	--
S1	Spalling of cover concrete, exposing no reinforcement	L_S
S2	Spalling exposing transverse reinforcement	L_S, L_T
S3	Spalling exposing longitudinal reinforcement	L_S
S4	Spalling exposing both transverse and longitudinal reinforcement	L_S, L_T

In order to measure the extent of spalling along the longitudinal axis of the column (length) the two dimensions, L_S and L_T, are calculated (Figure 4.3). The length of the spalled region, L_S, can be determined directly from the spalled map result at the summation of the spalling detection stage. As previously mentioned, the map represents the location of each spalled pixel in the image or video frame. A novel connected component labeling algorithm is then applied to these regions in order to find the pixel measurement for L_S. The connected component labeling algorithm shifts through every pixel in the image grouping each set of "detected" pixels which border one another into distinct regions. Connection is based on 8-connected objects, and the result is an integer matrix representation of the spalled map with each element index corresponding to the pixel location in the image and

value equal to the distinct region number. From this information, the region with the largest area (greatest number of pixels) is determined, and this region is considered as the spalled area of interest for this length measurement. A bounding box oriented in the same direction as the detected column is then created surrounding this region, and the length of this bounding box in the direction parallel to the column's longitudinal axis is considered the pixel representation of the L_S measurement. The relative measurement for the spalled length is also calculated with respect to the pixel representation of the width of the column. The length of between extreme exposed transverse reinforcing bars, L_T, is calculated from the segments resulting from the transverse reinforcement detection algorithm. For each segment detected, a bounding box is created around the entire region of pixels. Then, each box/segment is sorted according to the vertical coordinate of the center of the box. Since the image is oriented according to the axes of the detected column, the difference between the vertical coordinates of the first and last segments in the list is the pixel measurement for L_T. The relative measurement for the length between extreme exposed transverse reinforcing bars is also calculated with respect to the pixel representation of the width of the column.

4.4. Crack Property Retrieval

Several existing methods in crack detection on concrete surfaces have already been discussed (Section 3.2.1). The adaptation of the percolation-based method proposed by Yamaguchi and Hashimoto (2010) used in the method for transverse reinforcement detection in this work is also employed for crack detection in this research due to its robustness to image noise and fast crack detection speed on

large-scale concrete surfaces. The method differs from that employed in the detection of transverse detection in a couple of ways. The percolation, region-growing detection procedure is initiated at crack boundaries which are characterized with high gradient magnitudes of crack and non-crack pixels to manage the percolation process. The method in crack detection utilizes the intensities in each of the Red, Green and Blue channels of the RGB image rather than the blurred CMYK image channels used in the transverse reinforcement detection method presented in Section 4.3.1.2. In addition, the circularity, F_c of the detected region, D_p is measured according to Equation 4.5, but the threshold value is set at 0.15, such that all pixels in the region D_p with circularity estimates less than 0.15 are considered crack pixels. This value is lower than that used in the detection of transverse reinforcement (0.18) due to the thinner nature of cracks in comparison to transverse reinforcement. In addition, the orientation of the regions is not taken into consideration at this stage in the algorithm since cracking can occur at any orientation on the column surface.

Once all the crack pixels are detected, the crack map is produced, and each crack in the crack map must be isolated in order to retrieve specific properties. The overview of the proposed method in crack property retrieval is shown in Figure 4.7. The properties extracted for this framework include crack length, orientation, maximum width, average width and spacing. With all of these properties, the crack pattern for the column surface is defined. In general, the process for segment isolation used in the transverse reinforcement detection algorithm is also used to isolate crack segments in the crack map. A combination of a binary image thinning

algorithm (Cychosz, 1994) and a Euclidean distance transform (Fabbri et al., 2008) is used to represent the crack property information.

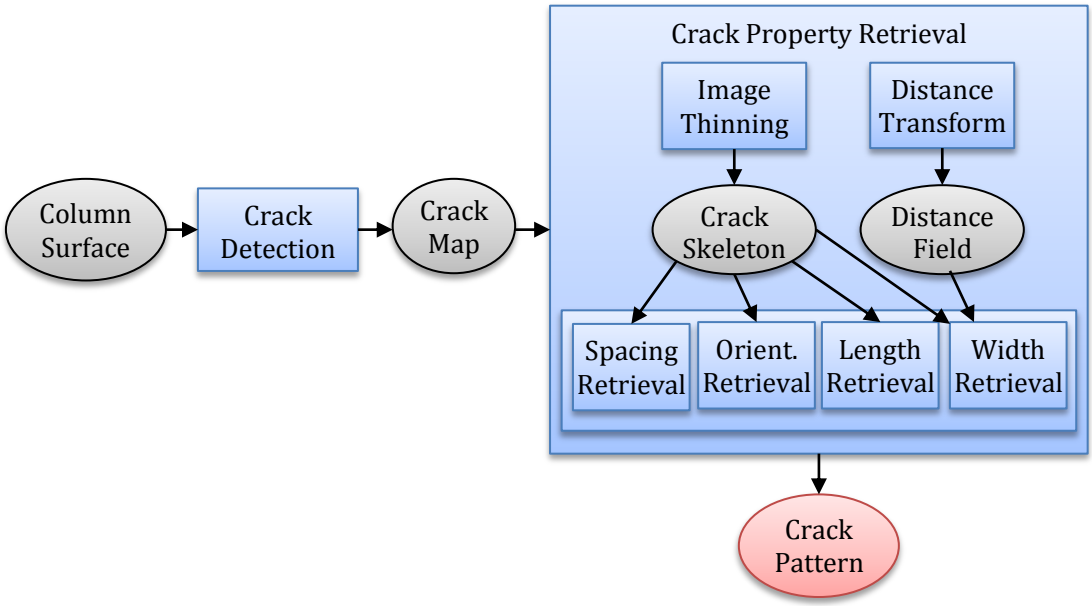
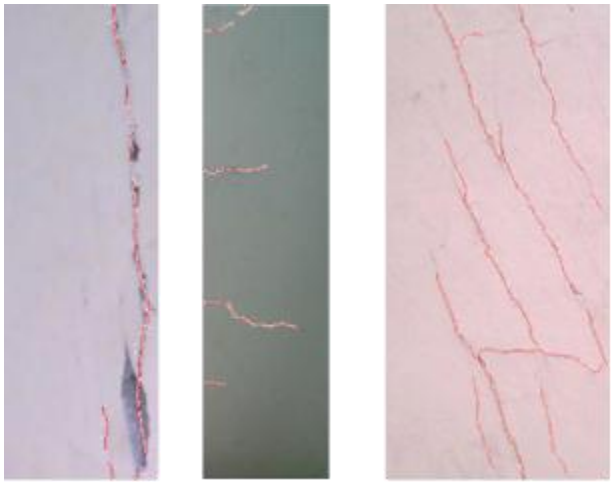


Figure 4.7: Overview of method in automated crack properties (pattern) retrieval

The topological configuration of a crack is ascertained by checking crack skeleton point connectivity using the segmentation algorithm discussed in Section 4.3.1.2 for the method in transverse reinforcement detection (Figure 4.6). However, as discussed in Section 4.3.1.2, with respect to cracks, the branching aspect of the segmentation algorithm is employed frequently. If there is more than one crack skeleton point connected to any one point, the current crack segment stops growing, and new segments are created (branching). The number of newly created segments depends on the number of neighboring crack skeleton points.

The properties of a crack are retrieved based on its skeleton segment information and distance field. The crack length is equivalent to the crack skeleton

segment length, which is approximated by the height of an object-oriented bounding box that circumscribes crack skeleton segment points. The crack orientation is the crack skeleton segment orientation, which is indicated by the direction of the object-oriented bounding box. Similarly, the double of the largest distance value that exists at skeleton points represents the crack's maximum width. The following measurements are calculated for cracks on RC columns: (1) the angle of crack direction in relevance to the columns' longitudinal axis; (2) the projection of the crack length on the columns' width; (3) the largest crack width in relevance to the columns' width; and (4) the spacing of crack patterns in relevance to the columns' width.



(a) (b) (c)

Figure 4.8: Common crack type examples: (a) vertical; (b) horizontal; and (c) shear

4.4.1. Crack Pattern Specification

In order to effectively determine the damage state of an RC column, the specific response mechanism must be determined. This will be discussed further in Chapter 5; however, in order to determine the response mechanism, the type of cracks which exist on the column surface must be defined. In the context of this work, the types of cracks which should be considered are the following (or a combination of the three types): (1) vertical (Figure 4.8(a)) – along the longitudinal axis of the column; (2) horizontal (Figure 4.8(b)) – perpendicular to the longitudinal axis of the column; and (3) shear (Figure 4.8(c)) – at approximately a 45° angle to the column axis. In this method, first the cracks are sorted according to their orientation. For this task, four separate List structures are created—two for each of the three crack types designated in Figure 4.8 (oriented to the left and oriented to the right: "Vertical-Left," "Vertical-Right," etc.). The List designation for each of the individual cracks is determined based on the orientation of the largest segment in the crack (deemed the dominant portion of the crack). Then, for each dominant crack in the lists, the spacing is calculated according to Equation 4.6 and Equation 4.7 using object oriented bounding boxes as is depicted in Figure 4.9. This figure also shows the meaning of each of the property measurements which are included within the context of the method presented in this research. The spacing is the distance between two (approximately) parallel cracks in the direction perpendicular to the orientation of the first crack. Thus, based on the orientation of the cracks in the image and the distance between each pair of consecutive cracks, the crack pattern is specified.

$$S_{1-2} = \left| y_{\text{int}_2} - y_{\text{int}_1} \right| \cdot \cos(90 - \theta_1) \quad \text{(Eq. 4.6)}$$

$$y_{\text{int}_i} = x_i - y_i \cdot \tan(\theta_i - 90) \quad \text{(Eq. 4.7)}$$

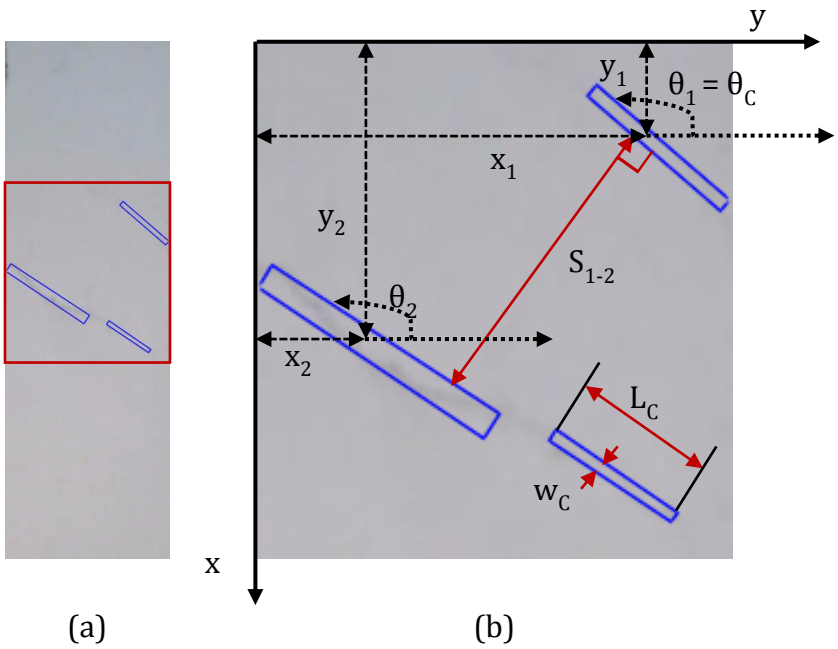


Figure 4.9: Example of crack properties retrieved and crack spacing calculation variable: (a) full image showing an entire column; and (b) close-up of boxed region

4.5. Implementation and Results

4.5.1. Validation

A comprehensive database was created for the purpose of this work. The database includes images of damaged and non-damaged reinforced concrete columns retrieved from various sources. Several of the images were retrieved on a reconnaissance trip to Port-au-Prince, Haiti in April of 2010. After the 7.0 M_W

earthquake on January 12, 2010 in Haiti, a team of researchers in the field of earthquake engineering, as well as two professional structural engineering consultants traveled to Haiti to retrieve manual and video data pertaining to various damaged RC frame buildings. The team focused on a total of five buildings, retrieving video "walk-thru" data for each column in the building (Figure 4.10(a)). In addition, the consultants focused on providing a detailed visual assessment beyond that routinely performed in post-earthquake safety or structural assessment procedures, but similar in nature to the desired, quantitative output for the automated procedures described in this dissertation (Figure 4.10(b)). Examples of these manual evaluations are presented in Appendix A. Thus, along with various other images of post-earthquake damaged and non-damaged columns (such as those in the NEES earthquake database (NEES, 2009)), the combination of these images and the manual evaluation surveys was used to validate the procedures discussed in this dissertation.

The performance of the methods in damage detection are evaluated using measures of precision (Equation 4.8), recall (Equation 4.9) and (1 – specificity) (Equation 4.10), where *TP* represents those pixels correctly detected, *FN* represents object pixels wrongly detected as background pixels (not detected), *TN* represents background pixels correctly detected as background pixels (not detected) and *FP* represents background pixels wrongly detected as object pixels. The performance of the methods in property retrieval is evaluated using the calculated percent error (Equation 4.11) and accuracy (Equation 4.12), depending on the nature of the

operation. The percent error is designated as the error between the actual (manually determined) values and the measured (automatically retrieved) values.

(a) (b)

Figure 4.10: Data collection in Haiti following 2010 earthquake: (a) video retrieval; and (b) manual evaluation

$$\mathrm{Pr}ecision = \frac{TP}{FP + TP} \qquad \text{(Eq. 4.8)}$$

$$\mathrm{Re}call = Sensitivity = \frac{TP}{TP + FN} \qquad \text{(Eq. 4.9)}$$

$$1 - Specificity = 1 - \frac{TN}{FP + TN} \quad \text{(Eq. 4.10)}$$

$$\%Error = \left(\frac{Actual - Measured}{Actual} \right) \cdot 100 \quad \text{(Eq. 4.11)}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad \text{(Eq. 4.12)}$$

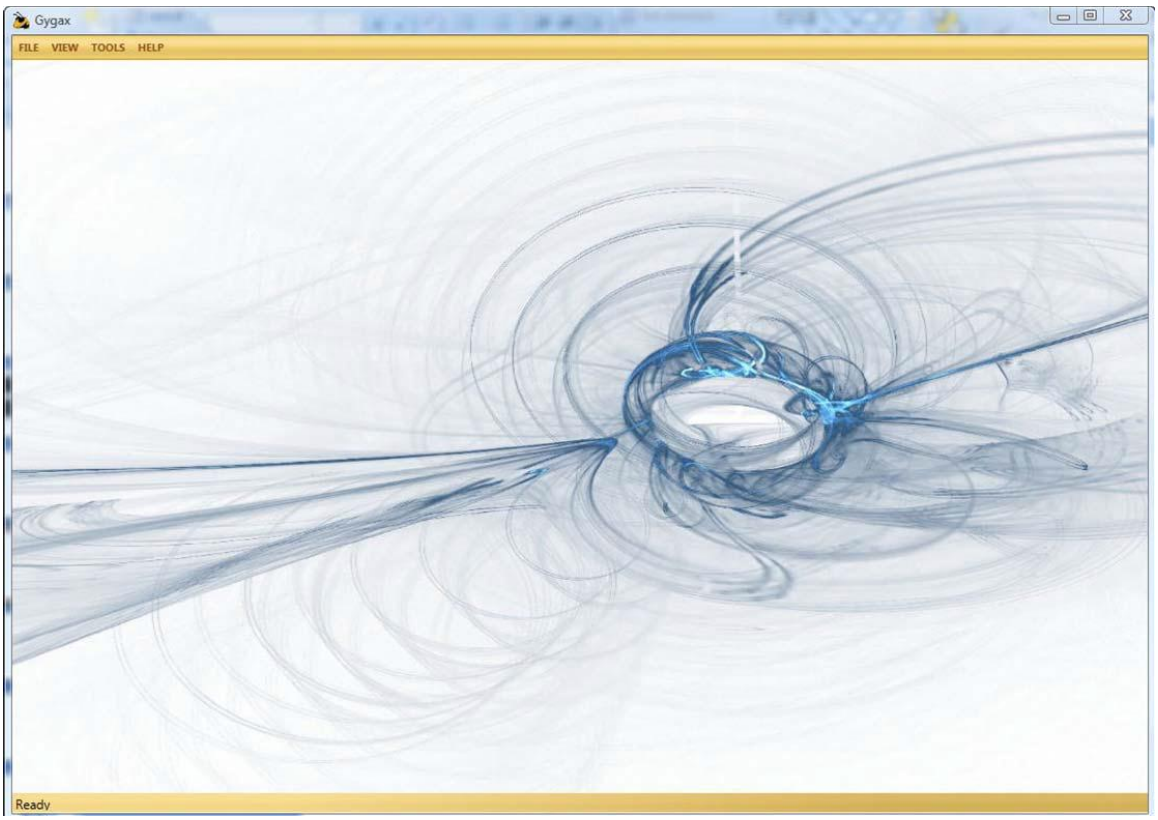


Figure 4.11: Graphical user interface main page for the prototype

4.5.2. Implementation

A prototype was developed to detect and retrieve the properties of the spalled regions and to retrieve the properties of cracking on concrete column surfaces using Microsoft Visual .NET, OpenCV and EmguCV. OpenCV is a collection of C functions and C++ classes. Many popular image processing and computer vision algorithms have previously been implemented in these collections. Therefore, it is used as the main image processing library for the spalled property retrieval prototype. EmguCV is used as a wrapper to enable the OpenCV functions to be used in the Microsoft Visual .NET environment. The methodologies were implemented and integrated into the larger prototype developed by the Construction Information Technology Laboratory, formerly at the Georgia Institute of Technology, as independent modules. The main graphical user interface (GUI) for the prototype is displayed in Figure 4.11.

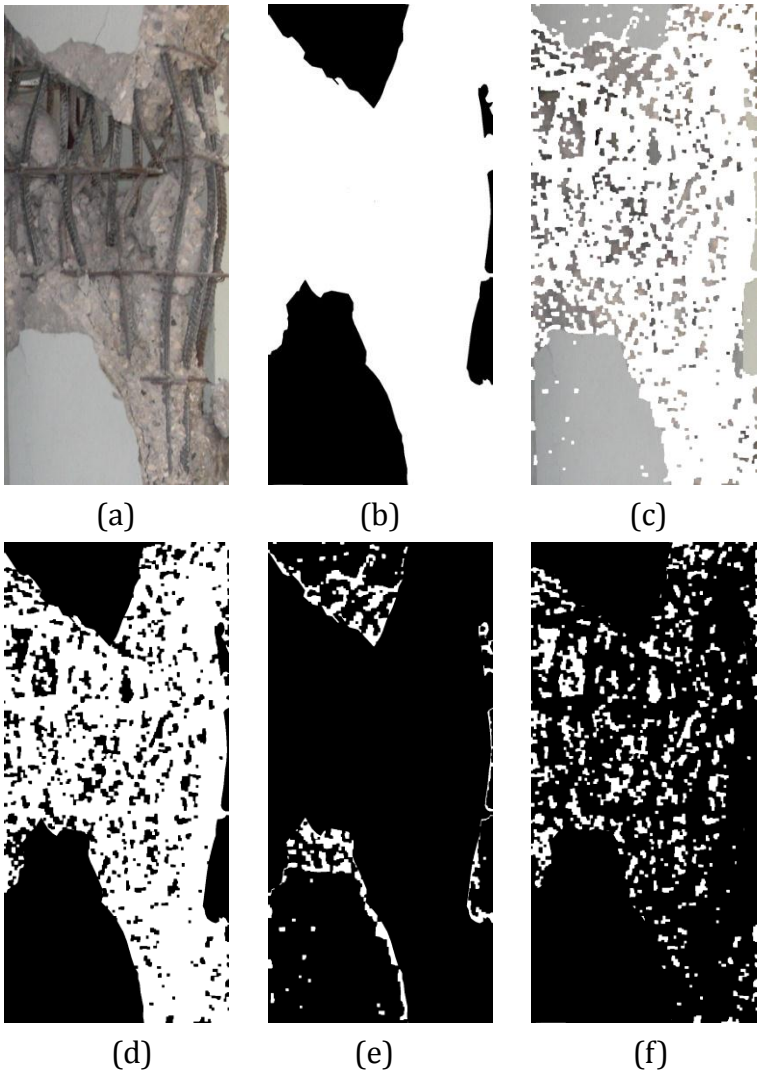


Figure 4.12: Spalling detection validation example: (a) original image; (b) ground truth; (c) detection map; (d) true positive; (e) false positive; and (f) false negative

4.5.3. Spalling Property Retrieval Performance

For the method in spalling detection and property retrieval, a database of 88 damaged RC column images was used. For the spalling detection portion of the method, the average precision (Equation 4.8) and recall (Equation 4.9) using a 9 x 9 neighborhood size for the local entropy-based threshold are 81.1% and 80.2%,

respectively. One example of the validation procedure for spalling detection is illustrated in Figure 4.12.

The performance of the method in spalled property retrieval is further measured with respect to the depth classifications and length measurements. The performance of the method in spalled depth retrieval (classification) is calculated using Equation 4.12, displaying the accuracy of the designation of a surface into one of the five categories specified in Table 4.1. For the 88 test images in the spalled image database, the overall accuracy for the classification of the depth of spalling on RC column surfaces was calculated as 87.53%. In addition, the accuracies for each individual class are calculated as 97.53% (S0), 93.18% (S1), 97.22% (S2), 70.97% (S3) and 77.55% (S4). Table 4.2 shows the results of the classification. In this table, the columns represent the number of sample images which were manually classified in each category and the rows represent the number of sample images which were classified into each category by way of the automated method discussed herein. Therefore, the sum of the values in each row is equal to the number of images of each class which was analyzed in the test, and the sum of the values in each column is equal to the number of images detected as each class. Finally, this means that the values in the *(i,j)th* cells where $i \neq j$ represent all of the false positive results (those images classified erroneously), and the values in the *(i,j)th* cells where $i = j$ represent all of the true positive results (those images classified correctly).

Table 4.2: Results for automated method in spalled depth classification

		Manually Classified				
		S0	**S1**	**S2**	**S3**	**S4**
Automatically Classified	**S0**	11	0	0	1	0
	S1	1	8	0	0	0
	S2	0	1	1	0	0
	S3	0	1	0	5	10
	S4	0	3	1	4	32
Accuracy		**97.53%**	**93.18%**	**97.22%**	**70.97%**	**77.55%**

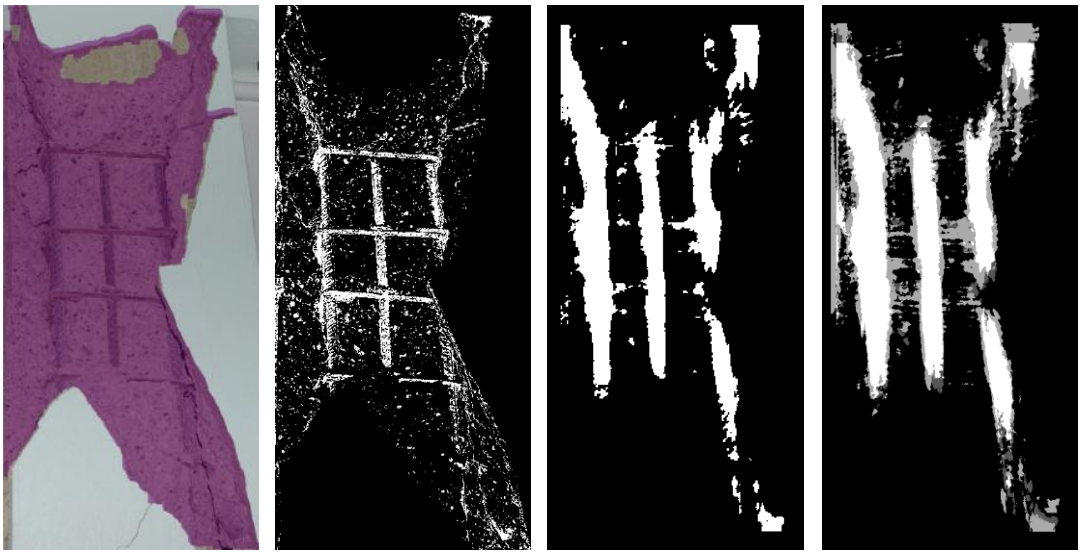
Table 4.3: Measurement error for length retrieval in 88 spalled images

	L_S / b	L_T / b
Average	4.22%	8.05%
Sdv	2.37%	5.74%

*L_S, L_T defined in Figure 4.3; b = structural element (column) width

In contrast, the performance of the methods in spalled length calculation is determined using Equation 4.11 where the manually measured distances (L_S and L_T) relative to the actual measured width of the column are compared with those retrieved automatically. For the same 88 test images, the average percent error and the standard deviation in the measurements are displayed in Table 4.3. Figure 4.13 and 4.14 show one example of the intermediate results in the spalled property retrieval method. In specific, Figure 4.13 shows the intermediate results for the method in longitudinal reinforcement for a sample image, and Figure 4.14 shows the

intermediate results for the method in transverse reinforcement for the same sample image. Particular areas of sensitivity in the method of spalling detection and property retrieval are discussed in the following sub-sections.



(a) (b) (c) (d)

Figure 4.13: Intermediate results for the method in automated longitudinal reinforcement detection: (a) spalled map; (b) thresholded image (Cyan channel); (c) matched image; (d) combination of matched images

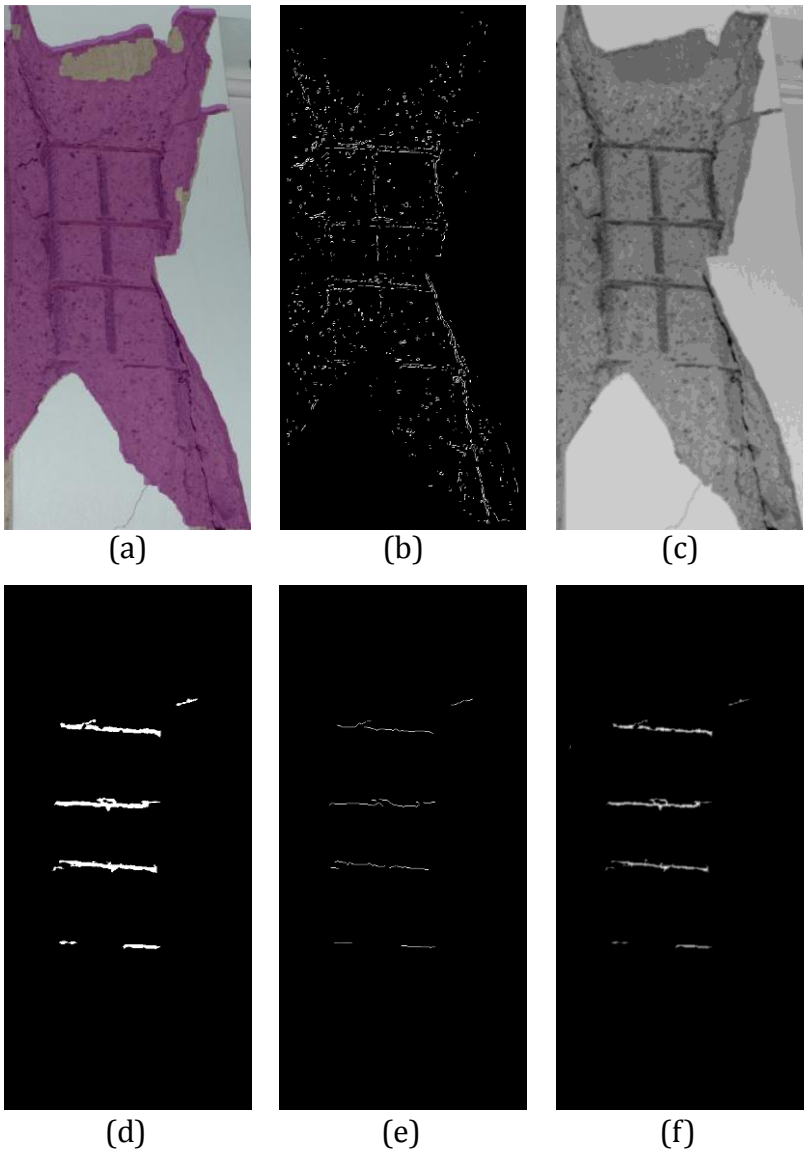


Figure 4.14: Intermediate results for the method in automated transverse reinforcement detection: (a) spalled map; (b) edge map; (c) smoothed image (Cyan channel); (d) transverse reinforcement map; (e) skeleton; and (f) distance map

4.5.3.1. Neighborhood Size Estimation

Multiple neighborhood sizes were implemented into the program in order to determine the appropriate size for the entropy calculation in spalling detection. The neighborhood size must be odd in each dimension such that the pixel of interest is

represented by the center element in the neighborhood array. In order to determine the optimum neighborhood dimensions, the precision (Equation 4.8) and recall (Equation 4.9) for the detection process are calculated with respect to the manual detection of the damage region of the images, and the tabulated results are shown in Table 4.4 for five of these neighborhoods (N = 5 x 5, 7 x 7, 9 x 9, 15 x 15 and 33 x 33). In general, a larger neighborhood size is more computationally expensive, as the number of comparisons is significantly larger. Therefore, the best choice for neighborhood size should be the smallest which still yields accurate results. It is evident from the precision and recall values in Table 4.4 that the 9 x 9 neighborhood is the most efficient and accurate sized neighborhood for this application.

Table 4.4: Average precision and recall for spalled detection neighborhood comparison

Neighborhood Dimension	Average Precision	Average Recall
5 x 5	0.602	0.926
7 x 7	0.692	0.833
9 x 9	0.811	0.802
15 x 15	0.809	0.710
33 x 33	0.799	0.685

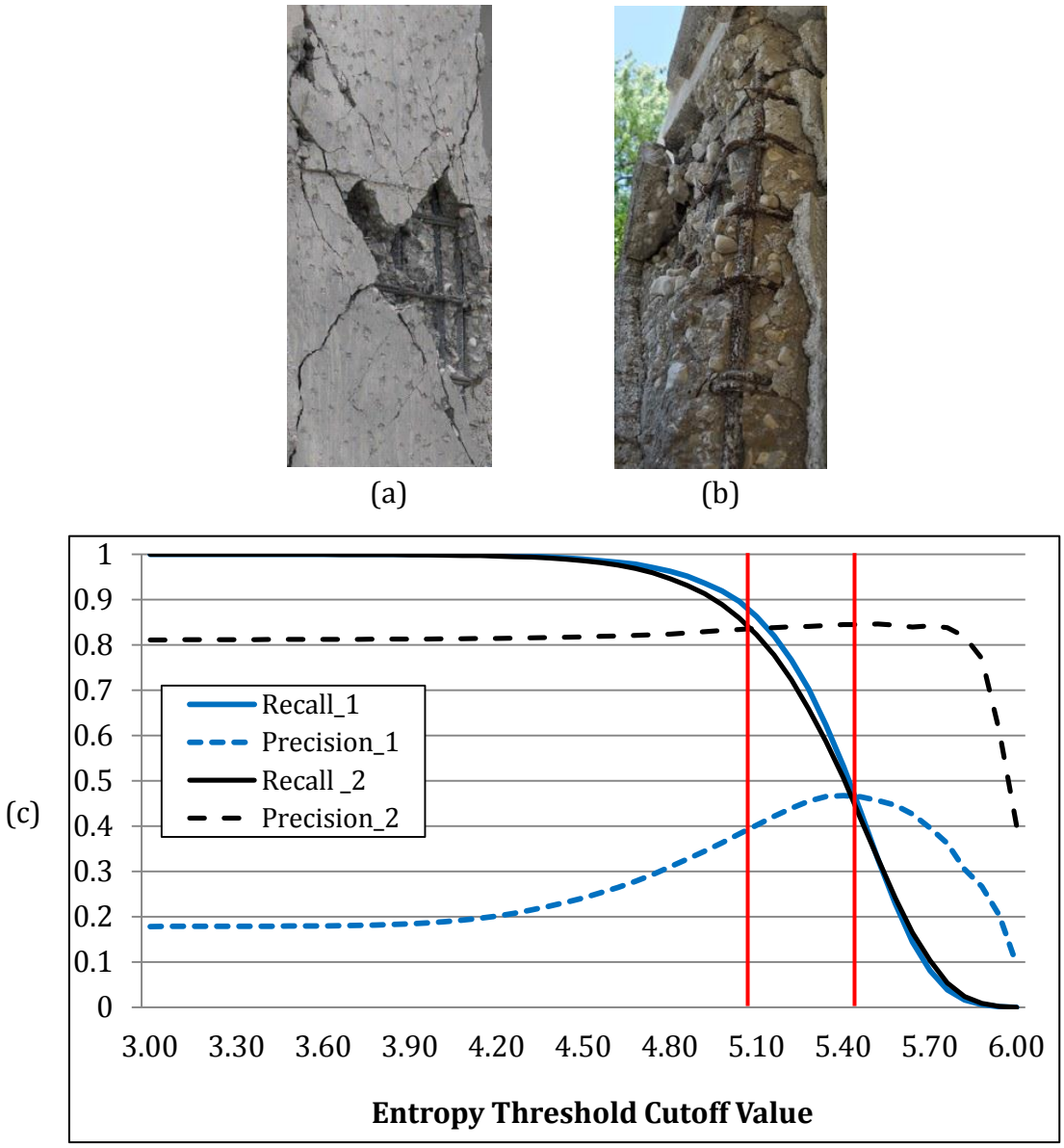


Figure 4.15: (a) Image 1 *(E_{avg} = 4.8649)*; (b) Image 2 *(E_{avg} = 5.1941)*; and (c) precision and recall curves for varying entropy cutoff values for images 1 and 2

4.5.3.2. Entropy Threshold Cutoff Analysis

The performance of the method in spalling detection is also highly dependent on the cutoff value used in the entropy thresholding operation. In order to measure the impact of this cutoff value in the detection procedure and thus, determine the

optimal cutoff value for the operation, the value is manually changed and the precision (Equation 4.8) and recall (Equation 4.9) are calculated for each resulting image. This operation is performed for every image in the database. In Figure 4.15 the precision and recall for ranging entropy cutoff values for two sample images are shown. From this figure, it is evident that the optimal cutoff value for the segmentation procedure is at the intersection of the two curves (along the red line) for any given image. This value varies for each image. However, by looking at the precision and recall curves for each individual image such as the examples displayed in Figure 4.15, a relationship between the mean and standard deviation of the overall entropy in the image and an optimal cutoff value can be estimated. Thus, based on this precision-recall analysis, a conservative value can be determined for each image, as a function of the mean and standard deviation values of the local image entropies.

4.5.3.3. Principal Color Model Identification

As was discussed in the background sections, detection in various color models has proved significantly more effective than that in the gray-scale depending on the aim of the detection. Due to the distinct color of steel, it is hypothesized that a color model other than the typical gray-scale or RGB model may prove superior. For this reason, a test set of images was analyzed in order to determine the particular color model which would provide the most accurate detection results. The five major color models (CIE, RGB, YUV, HSV and CMYK) and numerous sub-divisions of these modesl were analyzed for their effectiveness in reinforcement detection. Both an ROC (Receiver Operator Characteristic) analysis (Masumoto et al., 2000; Van Erkel

and Pattynama, 1998) and a precision-recall analysis (Van Rijsbergen, 1979) are performed on the image dataset. In the ROC analysis, the sensitivity (Equation 4.9) and (1 – specificity) (Equation 4.10) are compared for each distinct threshold in a binary segmentation system. In the precision-recall analysis, the recall (sensitivity) (Equation 4.9) and the precision (Equation 4.8) of the same process is plotted. Two representative plots are shown in Figure 4.16. For the ROC analysis, it is preferable for the curve to reside in the upper left corner, whereas, for the precision-recall analysis, the curve should reside in the upper-right corner. Overall however, the best case is when the area under the curve, in both cases, is equal to one. Thus, the area under the curve generated for each image was calculated. The average values are shown in Table 4.5. From the representative curves in Figure 4.16 and the average areas displayed in Table 4.5, it is evident that the CMYK color model is the most effective at distinguishing the object of detection (reinforcement) from its surroundings (spalled and non-spalled concrete).

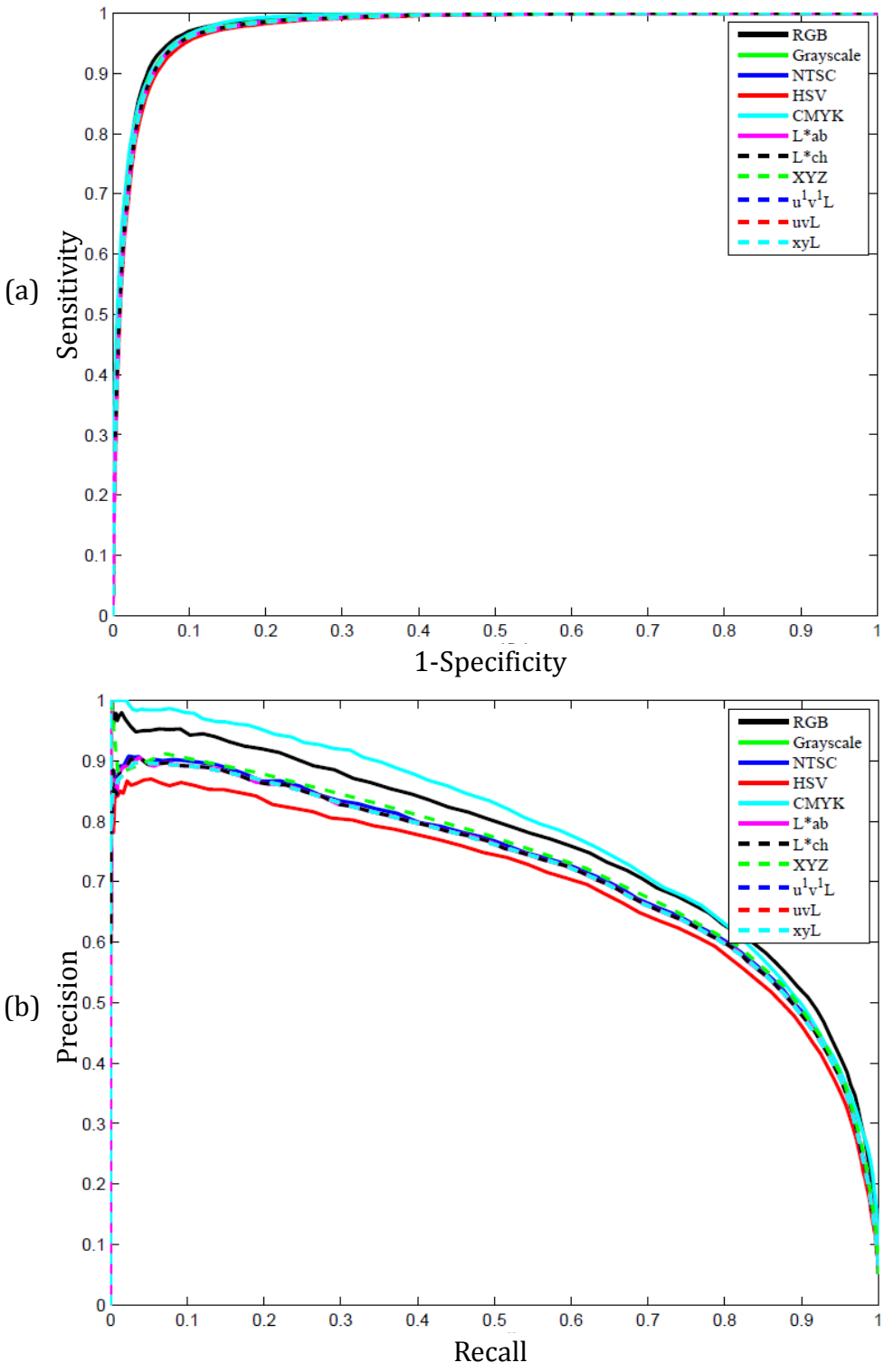
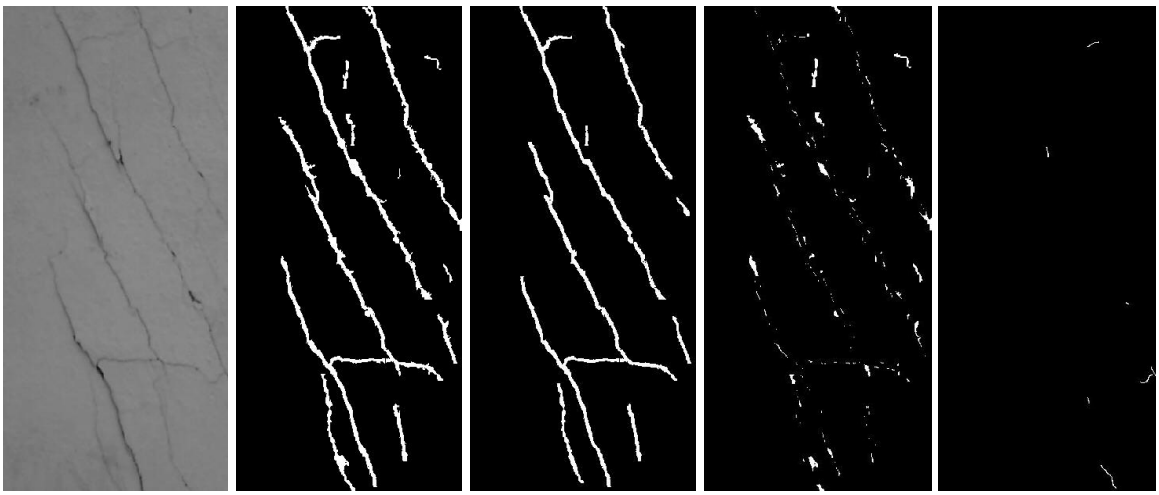


Figure 4.16: Representative performance curves for color model analysis: (a) ROC curve; (b) P-R curve

Table 4.5: Average area under performance curves for color model analysis of 40 sample images

Color Model	Average Area under ROC Curve	Average Area under P-R Curve
RGB	0.7853	0.3802
Grayscale	0.7812	0.3664
NTSC	0.7795	0.3684
HSV	0.7725	0.3795
CMYK	0.7932	0.4161
L*ab	0.7670	0.3855
L*ch	0.7763	0.3939
XYZ	0.7815	0.3500
u'v'L'	0.7602	0.3810
uvL	0.7600	0.3808
xyL	0.7663	0.3859



(a) (b) (c) (d) (e)

Figure 4.17: Crack detection validation example: (a) original image; (b) detection map; (c) true positive; (d) false positive; and (e) false negative

4.5.4. Crack Property Retrieval Performance

For the method in crack detection and property retrieval, a database of 100 damaged RC column images was used. For the crack detection portion of the method, the average precision (Equation 4.8) and recall (Equation 4.9) for the set of 100 images are calculated as 64.2% and 91.8%, respectively. One example of the validation procedure for crack detection is illustrated in Figure 4.17.

The performance of the method in crack property retrieval is further measured with respect to all of those cracks which were correctly detected. This is calculated in two ways: (1) the percent error (Equation 4.11) between the individual automatically retrieved property measurements and the manually retrieved measurements; and (2) the accuracy (Equation 4.12) in the classification of the crack pattern type. The first performance measure, representing the error in the individual measurements of each property (width, length, orientation and spacing), is displayed in Table 4.6. The average errors in automatically measuring these crack properties (in relation to the structural element) are 3.29° (orientation), 2.21% (length), 0.35% (maximum width) and 6.06% (spacing). In Table 4.7, the error in the relative automated spacing measurement is broken down according to each type of crack. Figure 4.18 shows one example of the intermediate results throughout the method in automated crack property retrieval. In addition, the individual measurements retrieved for all of the crack segments detected in a single image are displayed in Table 4.8.

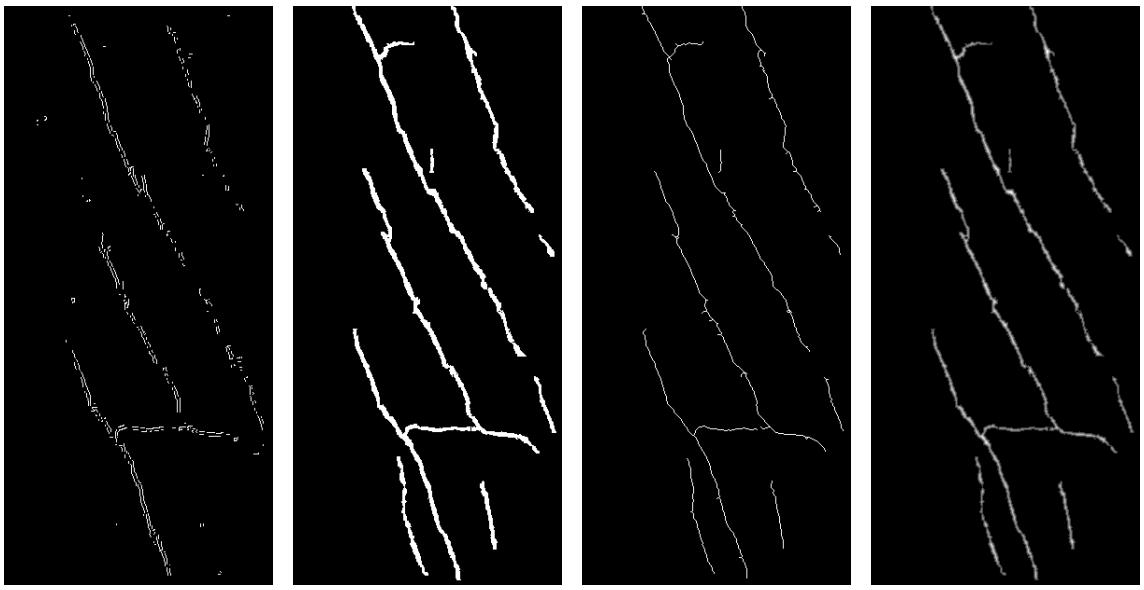
Table 4.6: Measurement error for property retrieval in 100 images/225 cracks

	Θ_C	L_C / b	W_C / b	S_C / b
Average	3.29°	2.21%	0.35%	6.06%
Sdv	2.70°	2.90%	0.49%	5.86%

*θ_C = angle WRT x-axis; L_C = crack length; W_C = maximum crack width; S_C = spacing between cracks of similar orientation; b = structural element (column) width

Table 4.7: Measurement error for spacing retrieval according to crack type

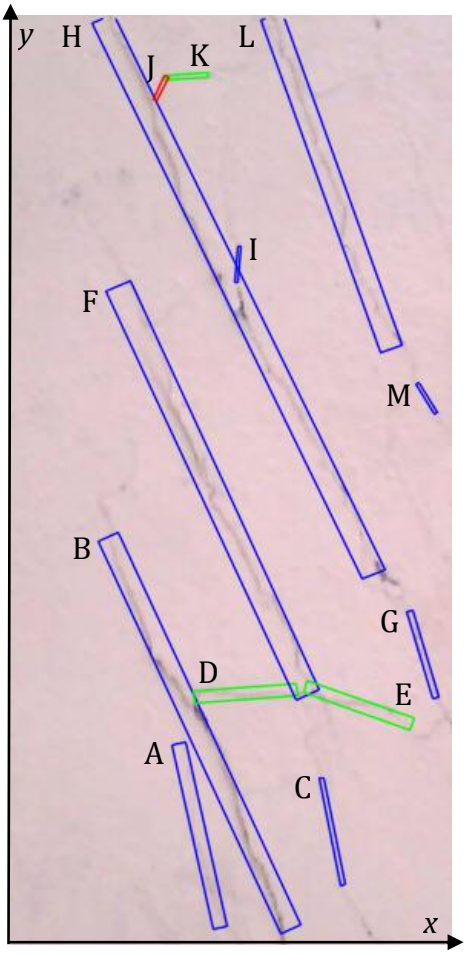
	Shear	**Vertical**	**Horizontal**
Average	5.72°	7.44%	5.01%
Sdv	6.17°	7.82%	0.49%



(a) (b) (c) (d)

Figure 4.18: Intermediate results in automated crack property retrieval: (a) edge map; (b) crack map; (c) crack skeleton; and (d) distance map

Table 4.8: Crack pattern measurements for segments A-M shown in figure on left



Crack Segment	Θ_C	L_C / b	W_C / b
A	102.583°	0.439	6.559
B	114.444°	0.999	0.023
C	101.310°	0.253	0.026
D	4.268°	0.233	0.023
E	160.292°	0.256	0.026
F	114.283°	0.999	0.026
G	106.504°	0.209	0.019
H	115.489°	1.427	0.027
I	85.236°	0.084	0.013
J	64.983°	0.061	0.027
K	3.814°	0.101	0.027
L	121.430°	0.081	0.026
M	109.458°	0.817	0.026
DOMINANT CRACK PATTERN : SHEAR			

*θ_C = angle WRT x-axis; L_C = crack length; W_C = maximum crack width; b = structural element (column) width

The complete performance for the method in automated crack pattern classification can be estimated in the same manner as the method in automated spalled depth classification. The overall intent of this method is to automatically determine the dominant crack pattern on the RC column surface (e.g., shear, vertical, horizontal or a combination of two types). Table 4.9 shows the results of this classification for the same 100 images used to estimate the error associated with each of the individual measured properties. Similar to Table 4.2 which shows

the results of the method in automated spalled depth classification, the columns in Table 4.9 represent the number of sample images which were manually classified into each crack type and the rows represent the number of sample images which were classified into each crack type by way of the automated method. Therefore, the sum of the values in each row is equal to the number of images of each dominant crack type which was analyzed in the test, and the sum of the values in each column is equal to the number of images detected with each dominant crack type. Finally, this means that the values in the *(i,j)th* cells where $i \neq j$ represent all of the false positive results (those images classified erroneously), and the values in the *(i,j)th* cells where $i = j$ represent all of the true positive results (those images classified correctly). Finally, the accuracy for the method in automated crack pattern classification can be calculated individually – for each crack type (shear: 92.93%; vertical: 91.92%; horizontal: 97.98%), and for the overall classification procedure (94.28%).

Table 4.9: Results for automated method in crack pattern classification

		Manually Classified		
		Shear	**Vertical**	**Horizontal**
Automatically Classified	**Shear**	31	6	1
	Vertical	1	50	1
	Horizontal	1	1	9
Accuracy		**92.93%**	**91.92%**	**97.98%**

4.6. Sensitivity Analysis

In computer vision applications, as in the real-world, it is not probable that the environmental, scene, lighting or other conditions will be ideal. Hence, in computer vision or image processing applications, several deviant scene condition characterizations should be considered. In this research, the conditions which are investigated are the following: (1) illumination; (2) blurring; (3) camera shift; (4) occlusion; and (5) scale variation. For the best possible detection results, an algorithm should be invariant to changes in all of these categories. However, this is not often possible, and so the algorithm should be defined such that the dependencies are of optimal nature for the specific application. Thus, the performance of each of these approaches in spalled property retrieval and crack property retrieval is analyzed according to these five metrics.

4.6.1. Illumination

Illumination is a metric concerned with the amount and effect of lighting within the image. Depending on the level of illumination, it can be exceedingly more difficult to discern key details in an image such as lines, corners, textures, etc. Since lighting is not a controlled variable in the application area of evaluations which may take place during sunlight hours, within buildings which have lost electricity, etc., the sensitivity to illumination for each of these methods in automated damage property retrieval is evaluated. In order to measure the sensitivity to illumination of these methods, five levels of illumination changes (L1-L5) are imposed on each of the images in the respective databases. The intensity values of the default image (L3) were set such that the average intensity was 128 (based on the 8 bit intensity

images ranging from 0-255). For the two darker levels, a constant intensity value (40% (L2) and 80% (L1) of 128) was subtracted from every pixel in the default image. Likewise, for the two brighter levels, a constant intensity value (40% (L4) and 80% (L5) of 128) was added to every pixel in the default image. Additionally, the maximum and minimum values in an image cannot go beyond 255 and 0, respectively. This is not an exact replication of the effect of illumination in the field. However, the loss of information which occurs due to those intensity values which, in fact, do exceed 255 or fall below zero, but are cutoff due to the addition/subtraction procedure is an adequate depiction of the illumination phenomenon.

For both the method in automated spalled property retrieval and the method in automated crack property retrieval the sensitivity to illumination is measured by calculating the average precision and recall values for detection at each level of illumination for the images in the respective database. Examples of a spalled image and a cracked image modified according to each level of illumination are displayed in Figure 4.19 (a-e) and Figure 4.20 (a-e), respectively. Accordingly, the precision and recall curves for spalled detection and crack detection are presented in Figure 4.19 (f) and Figure 4.20 (f) under these illumination variations.

From the curves in Figure 4.19 (f), it is evident that, in low-light scenarios, the effectiveness of the method in automated spalled detection and property retrieval is slightly diminished as both the detection precision and recall decrease for the lowest illumination level (L1). However, in general the method is not sensitive in overly-lit conditions. This is a result of the fact that the spalled regions

are usually darker in nature. Hence, as the illumination decreases (darker overall image) the pixels which are lost are those in the spalled region, and as the illumination increases (brighter overall image), the pixels which are lost are those on the non-damaged concrete surface (unspalled region).

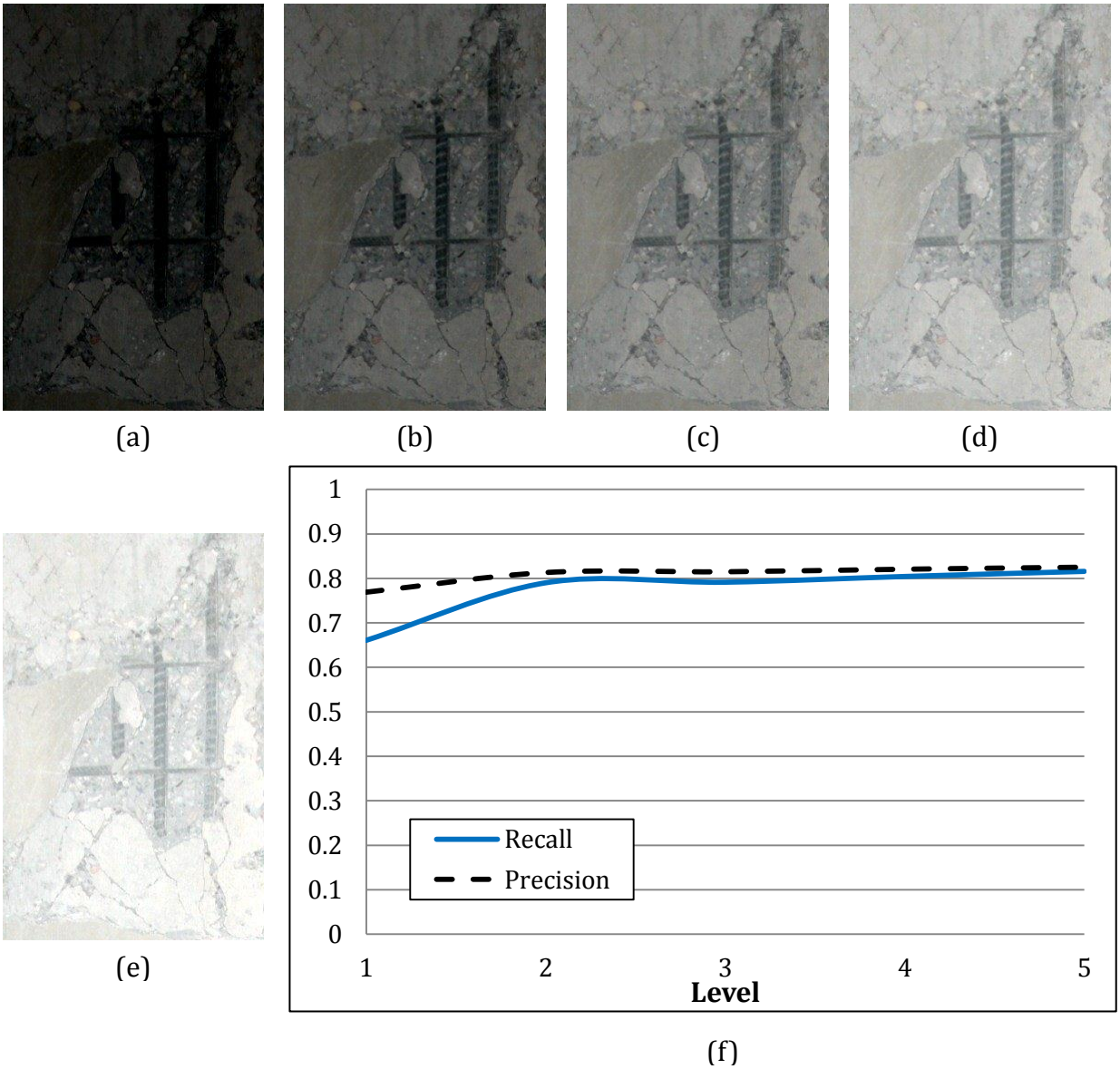


Figure 4.19: Results for the sensitivity analysis of the automated method in spalled property retrieval to illumination: (a) L1: -80% of baseline; (b) L2: -40% of baseline;

(c) L3: baseline; (d) L4: +40% of baseline; (e) +80% of baseline; and (f) average precision and recall curves for varied illumination

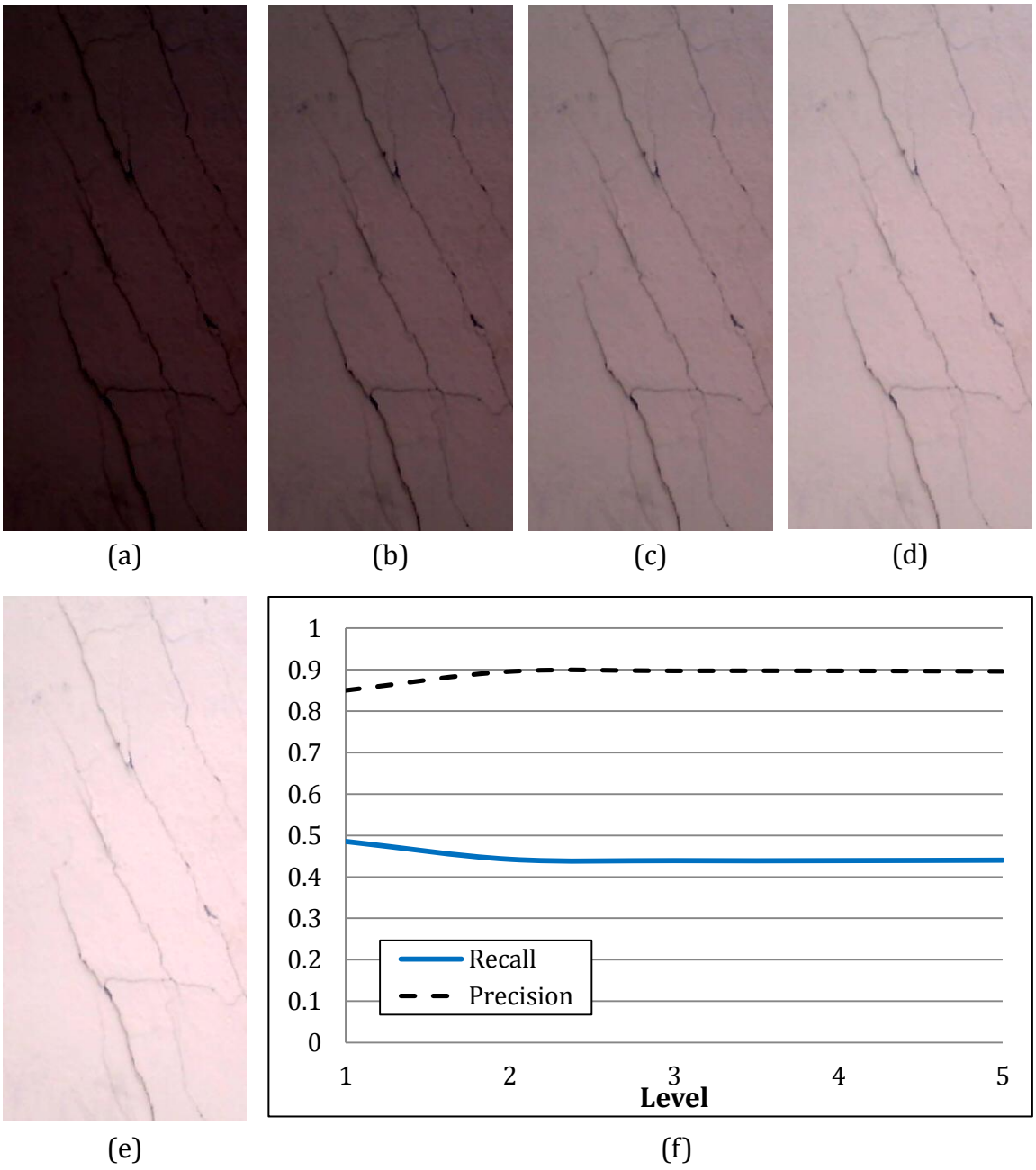


Figure 4.20: Results for the sensitivity analysis of the automated method in crack property retrieval to illumination: (a) L1: -80% of baseline; (b) L2: -40% of baseline; (c) L3: baseline; (d) L4: +40% of baseline; (e) +80% of baseline; and (f) average precision and recall curves for varied illumination

According to the curves in Figure 4.20 (f), the method in automated crack property retrieval is roughly invariant to changes in illumination. As discussed multiple times previously, the most dominant visual characteristic of cracking on RC element surfaces is the high gradient across image pixel intensities. Thus, these results are similar to those observed for the method in spalled property retrieval: as the illumination level increases, resulting in a brighter overall image, there will only be loss of the non-damaged RC element surface information. In addition, since the method employed for crack detection (percolation-based method), considers the connectivity of crack pixels and the intensity of the proposed crack regions relative to the proposed non-crack regions, even at the lowest level of illumination (L1), the cracks are still recognized, and the non-damaged concrete surface areas will remain unrecognized.

4.6.2. Blurring

The second metric considered in this sensitivity analysis is blurring. Blur is oftentimes present in an image or video frame as a result of misplaced focal points. In this way, when the focal point is identified as a point outside of the desired object of the detection (in the background), the background will be in focus and the foreground (desired object of detection) will be blurred. Due to the nature of the proposed application (rapid evaluation in hazardous environments) for this work, it is highly likely that several of the images and/or video data retrieved will be blurred. In order to simulate this effect and analyze the sensitivity of the methods in automated spalling and crack property retrieval, the original images in the databases are convolved with uniform disk filters of varying radii. The uniform disk

filter acts as an averaging mechanism for the pixel neighborhood designated by the filter radius. Similar to the other simulations of disturbance types in image and video data, this is not an exact replication of the effect of blurring which would be observed in the field.

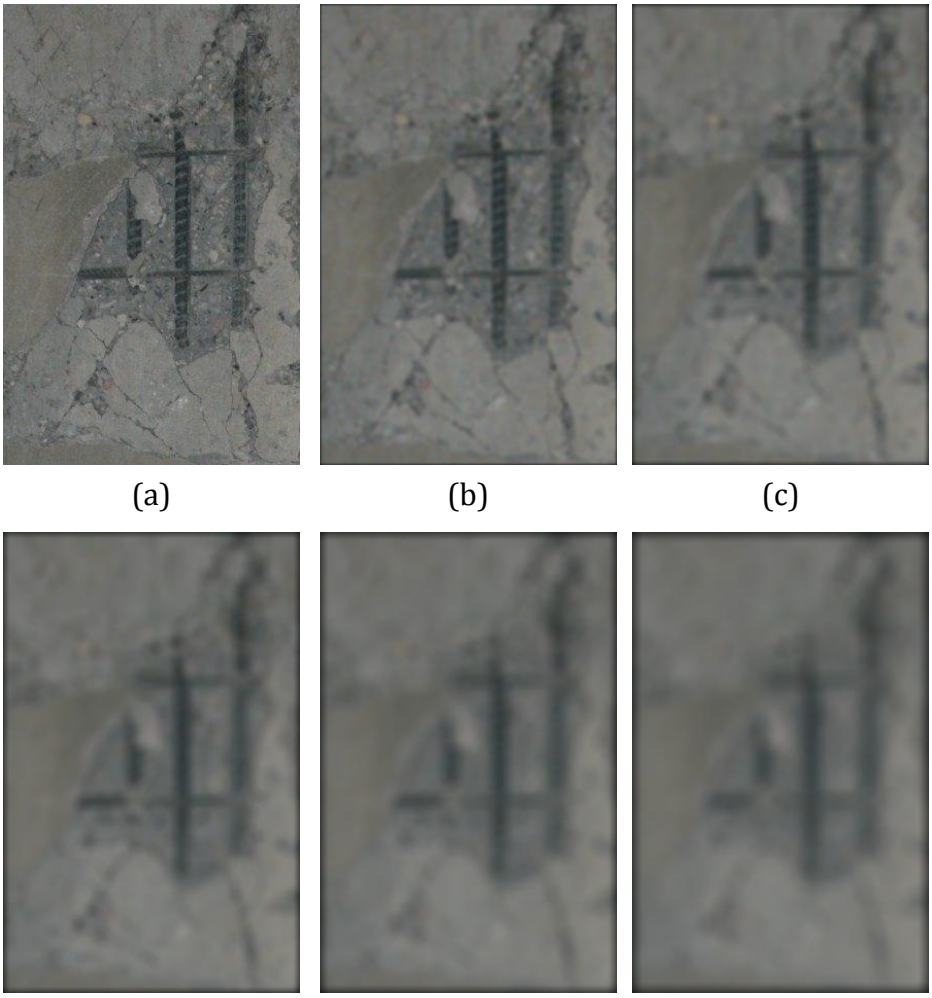


Figure 4.21: Sample results for induced blur in a spalled image: (a) original; (b) 3 x 3; (c) 5 x 5; (d) 7 x 7; (e) 9 x 9; and (f) 11 x 11

In Figure 4.21, an example spalled image (a) and the result of the original image convolved with the five filters varying in radius (3 x 3 (b), 5 x 5 (c), 7 x 7 (d), 9 x 9 (e) and 11 x 11 (f)) are shown. The average precision and recall across all of the images in the spalled image database were calculated and displayed in the form of the precision and recall curves in Figure 4.22. According to the degrading nature of both the precision and recall curve as the size of the filter increases, it is apparent that the method in automated spalled property retrieval is highly sensitive to blur. This is due to the local characteristic of the method: since the entropy thresholding algorithm operates on a neighborhood basis, the blurring of the image in each of those regions deems the operation nearly useless as all of the pixels in each neighborhood are becoming increasingly identical as the filter size increases to 9 x 9 (the size of the neighborhood used for the entropy threshold). At this point, both the recall and the precision level off to 0%. Both of these issues could be avoided by using a larger neighborhood size. However, as mentioned previously, the tradeoff for a larger neighborhood size is time; therefore, for the purposes of this work, where a rapid estimate is of utmost significance, the blur should be controlled by other (manual) means.

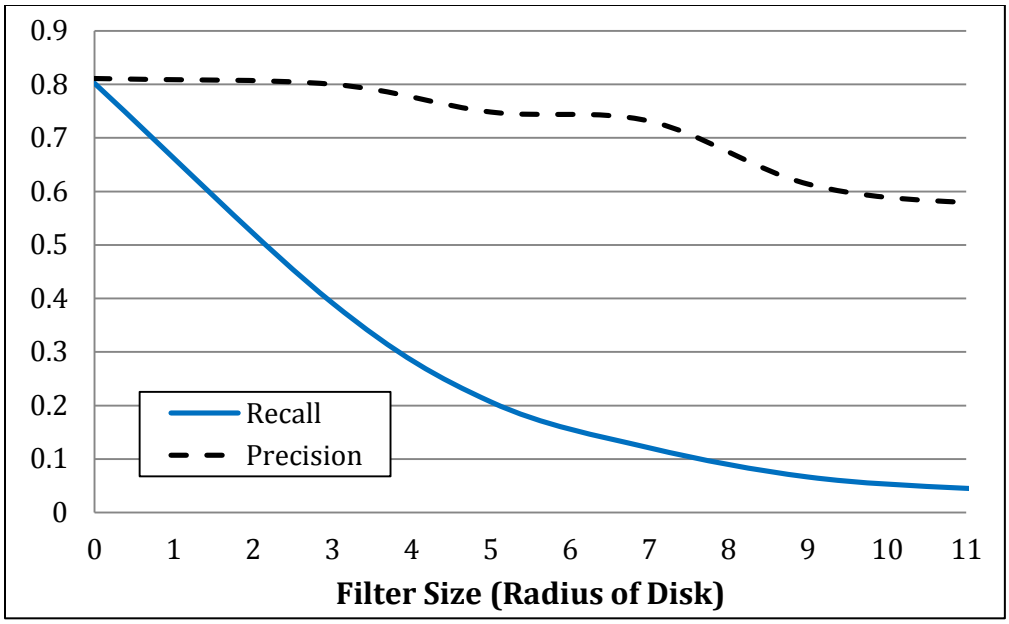


Figure 4.22: Average precision and recall curves for the sensitivity analysis of the automated method in spalled property retrieval to blur

For the analysis of sensitivity to blur for the crack property retrieval method, the resulting blurred images and precision/recall curves are displayed in the same manner as those for the method in spalled property retrieval in Figure 4.23 and Figure 4.24, respectively. In the recall curve shown in this figure, with the introduction of the 3 x 3 filter, the value immediately diminishes to near-zero. The depiction of a crack in an image is often only a few pixels wide; therefore, when the non-damaged surface area pixels and the crack pixels are not in focus, the number of pixels which should be detected as crack pixels will increase dramatically. Based simply on the ratio of the number of crack pixels to the number of concrete surface pixels typical in an image of a RC column, when the pixel values are averaged, the concrete surface pixels most often dominate and thus the number of pixels detected as cracks (falsely or correctly) will be minimal.

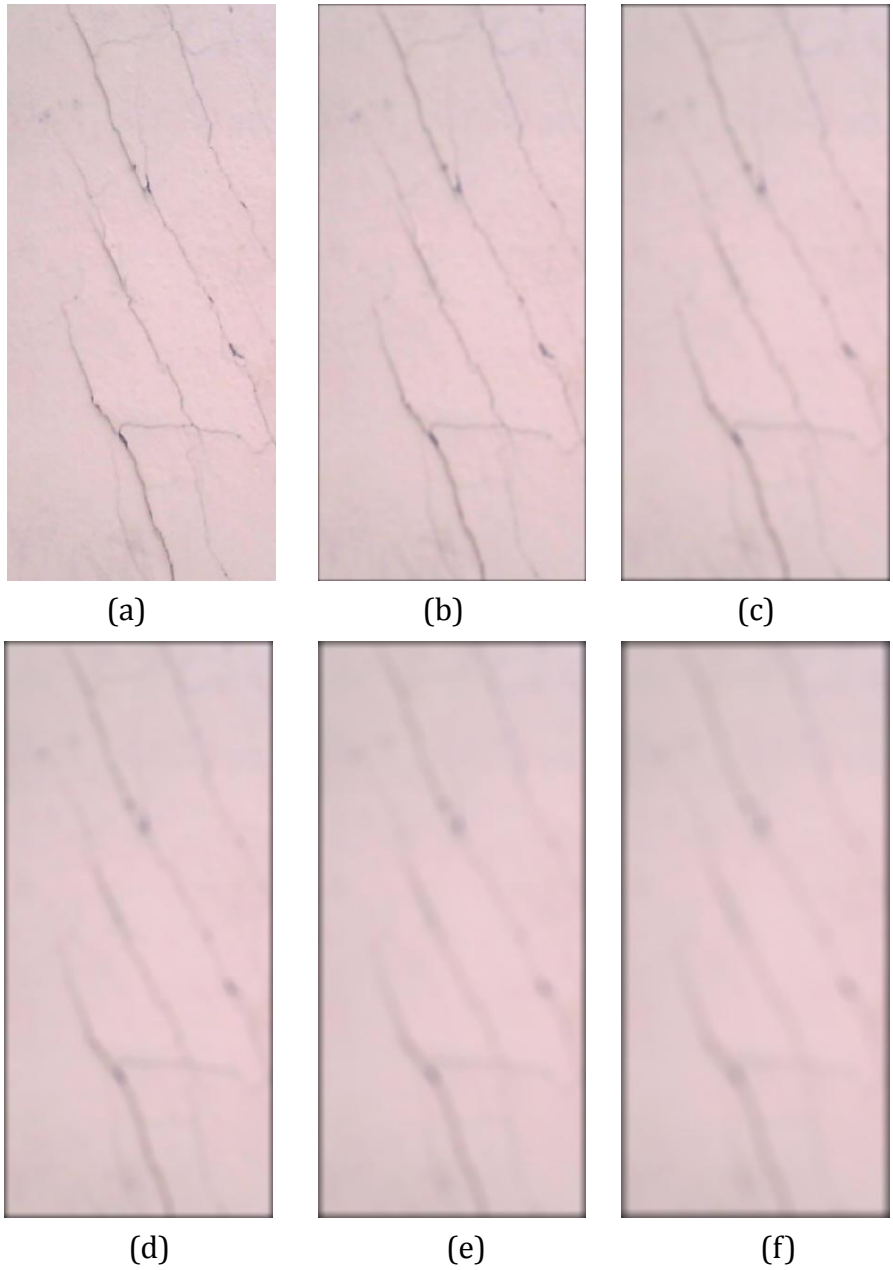


Figure 4.23: Sample results for induced blur in a cracked image: (a) original; (b) 3 x 3; (c) 5 x 5; (d) 7 x 7; (e) 9 x 9; and (f) 11 x 11

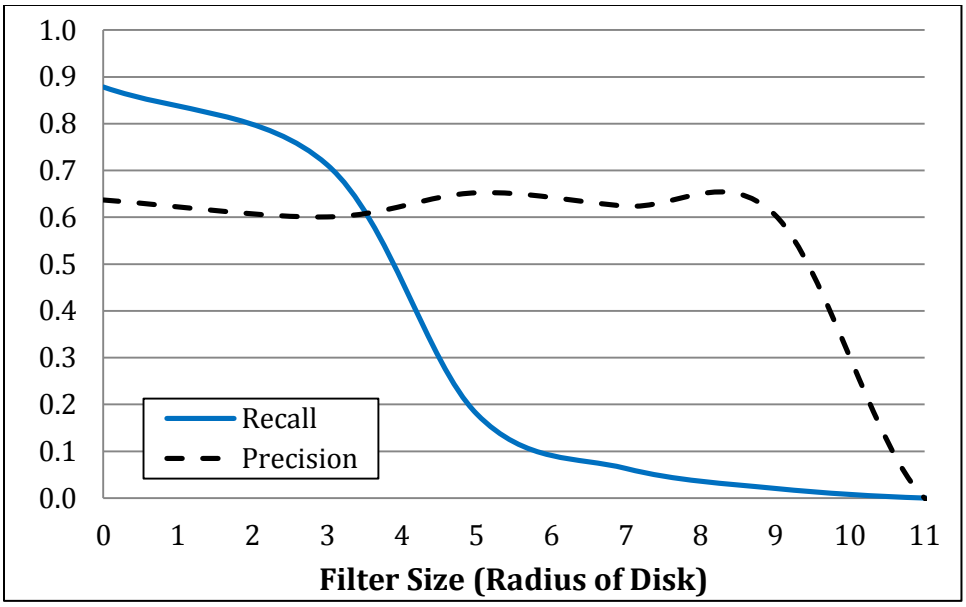


Figure 4.24: Average precision and recall curves for the sensitivity analysis of the automated method in crack property retrieval to blur

4.6.3. Camera Shift

In addition to blur caused by misplaced focal points, blur can also be introduced in an image due to a shift in the camera physically when the images are being captured. In specific, it can be the result of a slow shutter speed setting on a still camera or frame-to-frame shifting in video cameras. When camera shift is present, the image will appear blurred in the direction of the shift. In order to adequately determine the sensitivity of these methods to camera shift, the effect of shift-blurring is simulated with a range of linear digital filters. The filters range in orientation (0°, 45°, 90° and 135°) in order to account for the different orientations in which the camera might experience a shift. In addition, the filters range in the amount of shift (2, 4, 10 and 20 pixels). An example of the appearance of a 20 pixel shift in each orientation is displayed in Figure 4.25 for a sample spalled image.

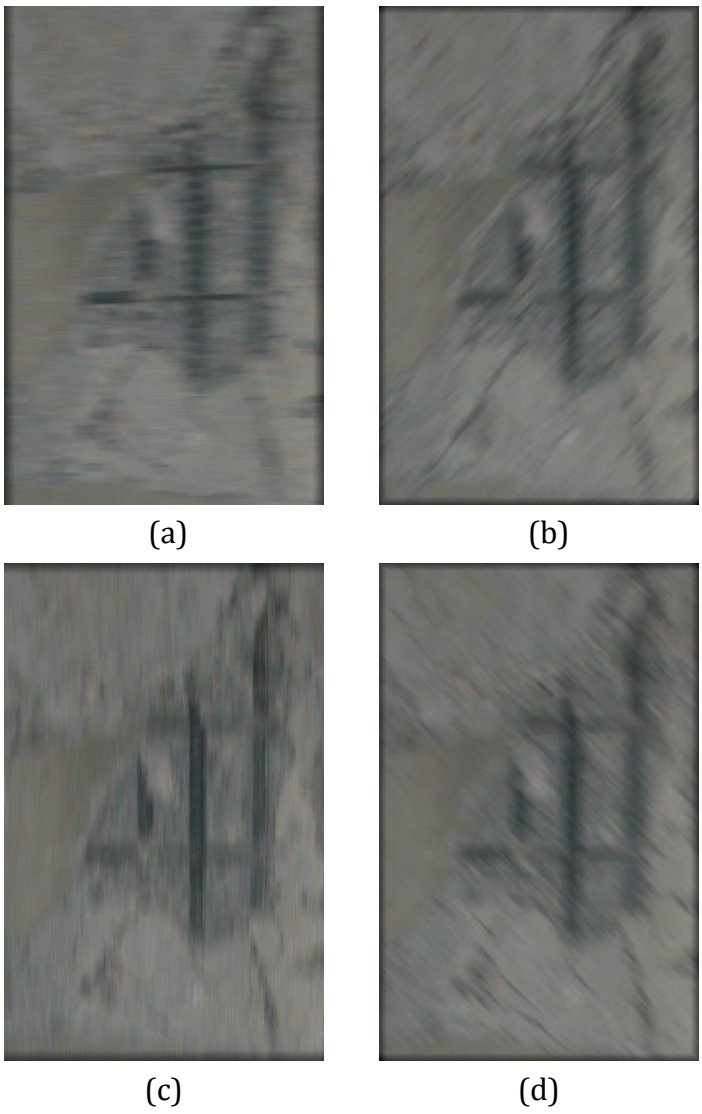


Figure 4.25: Example of spalled image with shifted 20 pixels at: (a) 0°; (b) 45°; (c) 90°; and (d) 135°

For each orientation, shift and image in the spalled database, the precision and recall values were measured. The average values for each orientation and shift amount are represented in Figure 4.26 for the method in automated spalled property retrieval. This figure reveals that, for all shift orientations, as the number of pixels in the shift increases, the recall decreases. The introduction of a shift in any

direction due to movement of the camera, even a small amount, increases the number of pixels falsely identified as spalled pixels, and as the shift value increases, the recall continues to diminish at a moderate rate. In contrast, the precision appears invariant to any shift in any direction. These observations are in-line with those noted in Section 4.6.2, with regards to blurring, and are sensible since the method in automated spalled property retrieval primarily considers the local entropy of the image which is modified such that all of the regions are chaotic, rather than just the spalled region.

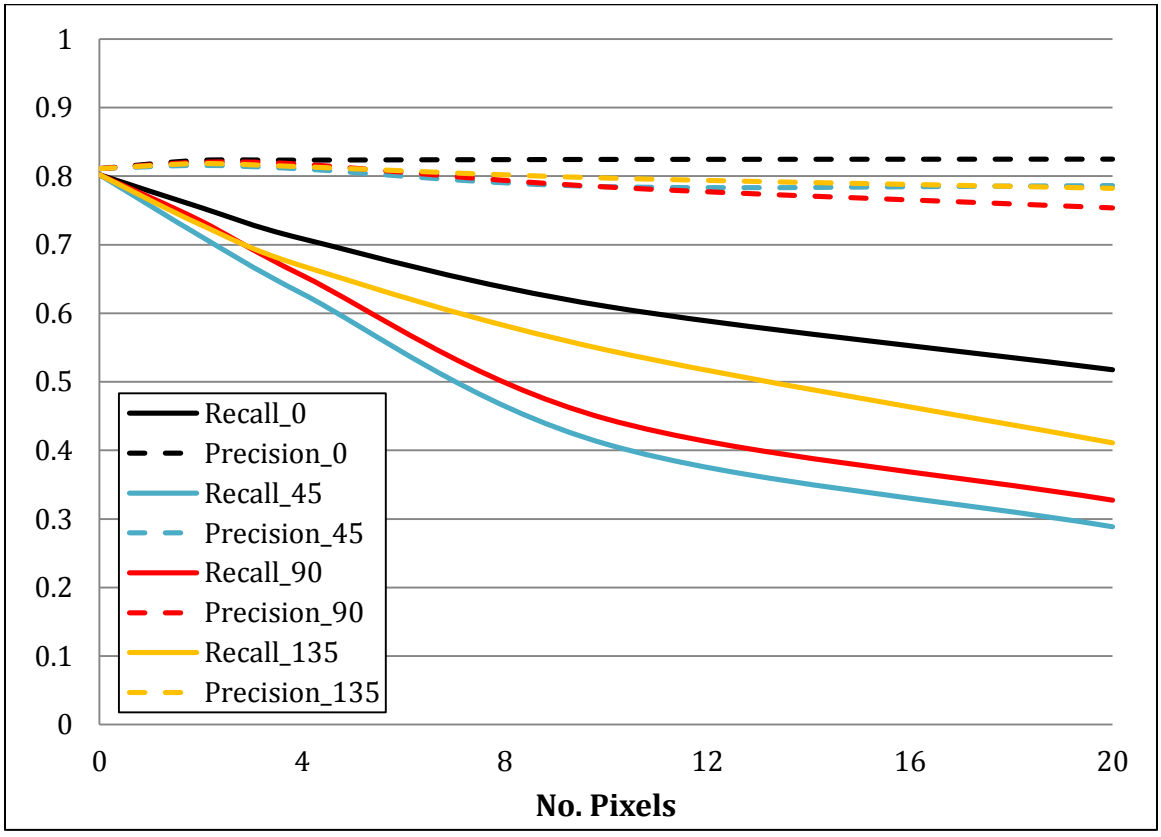
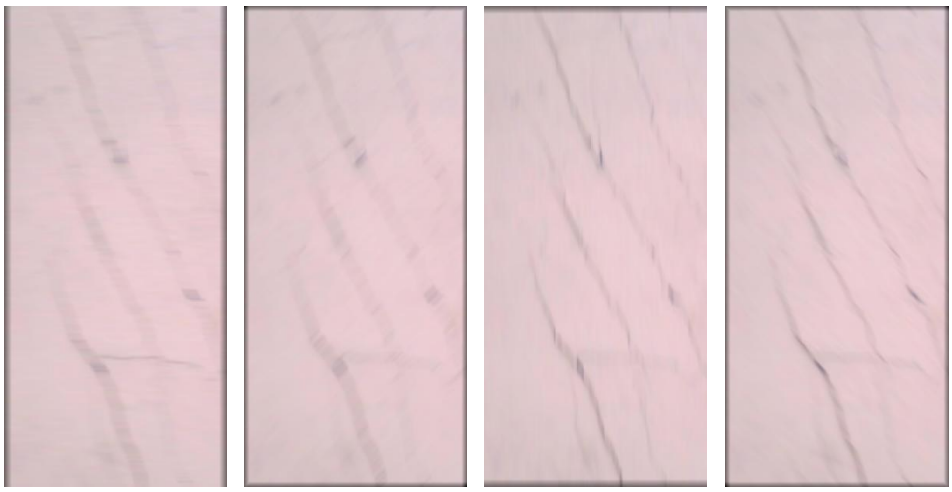


Figure 4.26: Average precision and recall curves for the sensitivity analysis of the automated method in spalled property retrieval to camera shift in four different directions

The same analysis is performed for the automated method in crack property retrieval. All of the images in the crack image database are shifted by each of the four pixel amounts, in each of the four orientations. A sample of the result of this shifting is shown for a single crack image in Figure 4.27. The precision and recall values are measured for each image and combination of shift variables, and the average of these values for each orientation are plotted against the number of pixels in the shift (Figure 4.28). These curves reveal that the method in crack property retrieval is most sensitive to shifts along the 45° axis (i.e. Figure 4.27 (b)). This is likely due to the fact that the crack image database has a greater number of images with shear cracks (at 135°). Thus, these results show that due to the linear nature of cracks, there will be more false negatives and thus a lower recall, when the shift is perpendicular to the direction of the crack.



(a) (b) (c) (d)

Figure 4.27: Example of cracked image with shifted 20 pixels at: (a) 0°; (b) 45°; (c) 90°; and (d) 135°

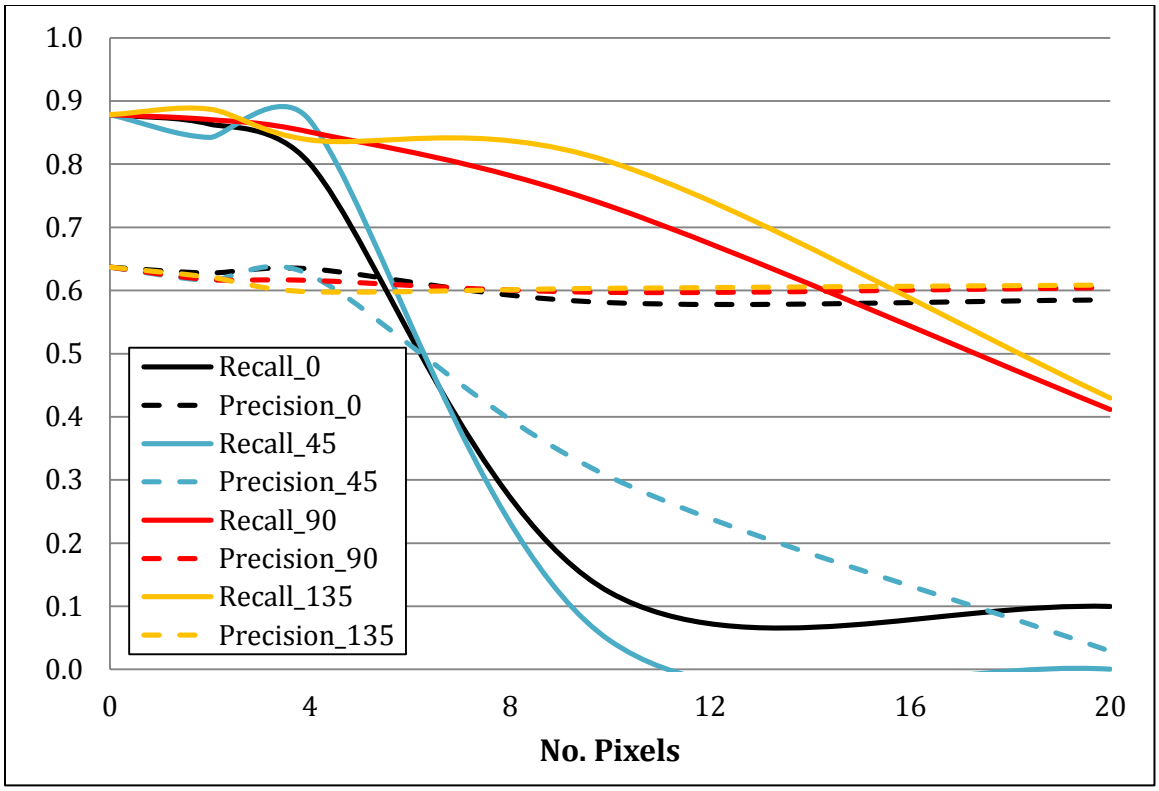


Figure 4.28: Average precision and recall curves for the sensitivity analysis of the automated method in crack property retrieval to camera shift in four different directions

4.6.4. Occlusion

In image processing problems, occlusion, the partial or full blocking of the desired object of detection, is often an issue. Specific to damaged buildings in a post-earthquake scenario, it is highly probable that part of the structural element, other structural elements or various non-structural elements may fall in front of the surface of the desired object in the image frame. By collecting video data, including the structural elements from several angles, the issue of occlusion is lessened. However, it is still necessary to test the algorithms in spalled and crack property retrieval with respect to their respective sensitivities to occlusion. In order to

simulate occlusion in the images, both horizontal and vertical blocks are independently introduced in increasing width across and down the image.

In Figure 4.29, the result of varying degrees of horizontal occlusion (10%, 20%, 30%, 40%, 50%, 75%) in a spalled image are shown, and in Figure 4.30 the same is shown for varying degrees of vertical occlusion of the same spalled image.

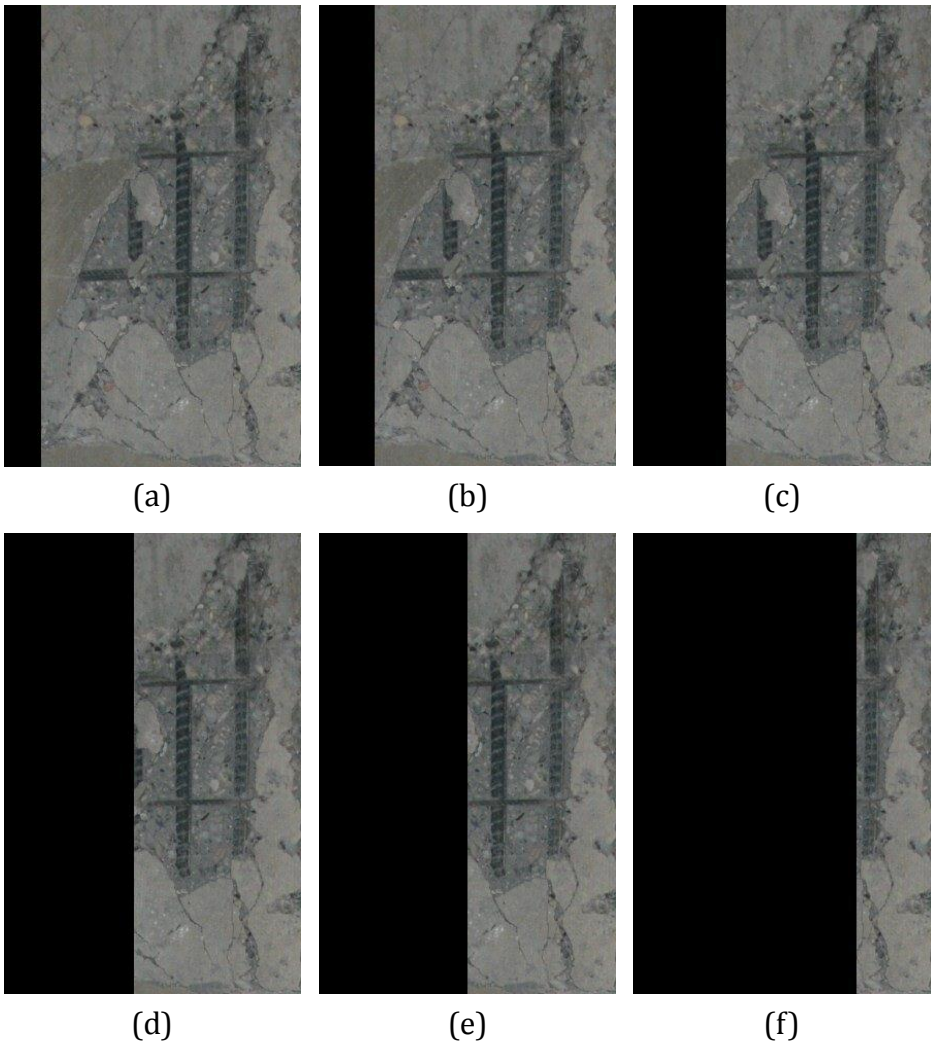


Figure 4.29: Sample results for manufactured horizontal occlusion in a spalled image: (a) 10%; (b) 20%; (c) 30%; (d) 40%; (e) 50%; and (f) 75%

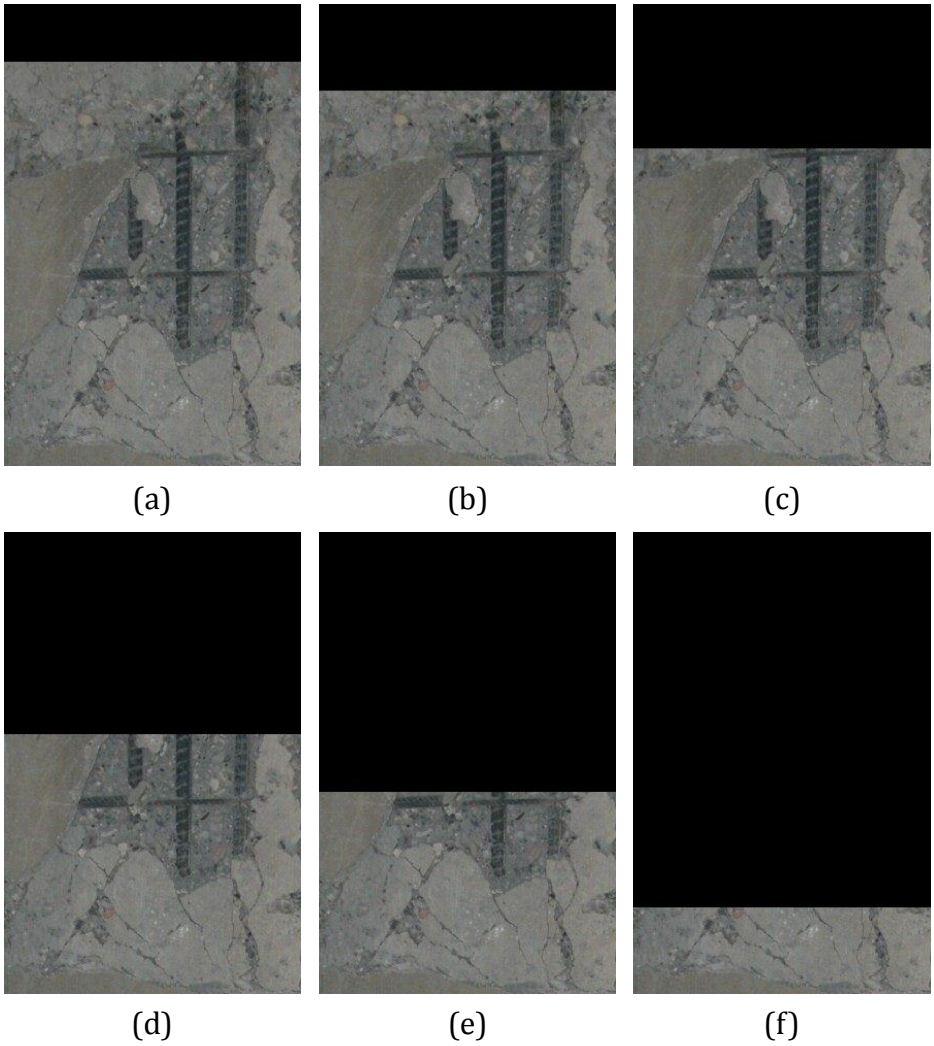


Figure 4.30: Sample results for manufactured vertical occlusion in a spalled image: (a) 10%; (b) 20%; (c) 30%; (d) 40%; (e) 50%; and (f) 75%

The sensitivity of the spalled property retrieval method to occlusion is measured by calculating the average precision and recall across all of the images in the spalled image database at each degree of occlusion. This is done for both horizontally occluded images and the vertically occluded images, and the values are displayed in the form of precision and recall curves (Figure 4.31). These curves reveal a standard relationship between detection precision and recall and occlusion.

As the amount of the image which is occluded increases, the recall decreases comparatively. Due to the inconsistent and irregular shape of spalled regions (non-linear, non-rectangular, etc.), the method employed is not shape-based, and thus, the effect of occlusion is less severe. However, the detection is only possible in portions of the image which are visible. Thus, the number of false negatives will increase immediately to those which are below the occluding object. The amount that the recall does decrease is directly related to the amount of the image which is occluded. The recall is not affected because the actual method in automated spalled property retrieval is not affected, so there is not an increase in false positive results.

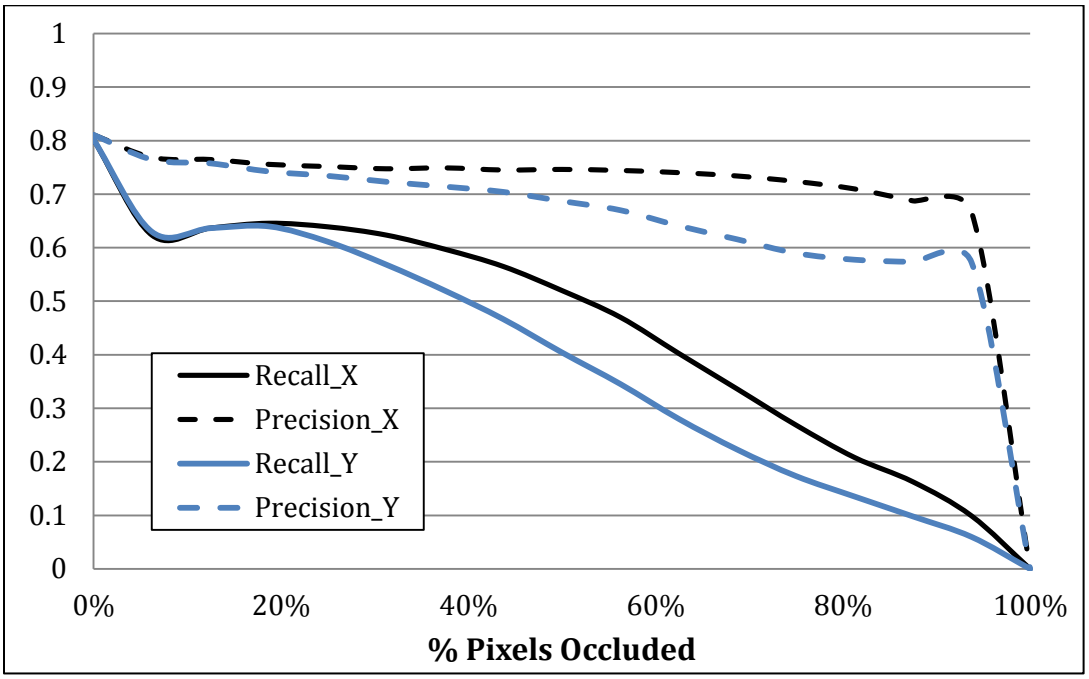


Figure 4.31: Average precision and recall curves for the sensitivity analysis of the automated method in spalled property retrieval to occlusion

In addition, the method in automated property retrieval is analyzed with respect to the level of sensitivity to occlusion. In Figure 4.32 (horizontal) and Figure 4.33 (vertical), the examples of a cracked image with varying degrees of occlusion (10%, 20%, 30%, 40%, 50% and 75%) are displayed. The average precision and recall values for each degree of occlusion and every image in the crack image database are calculated and depicted in Figure 4.34.

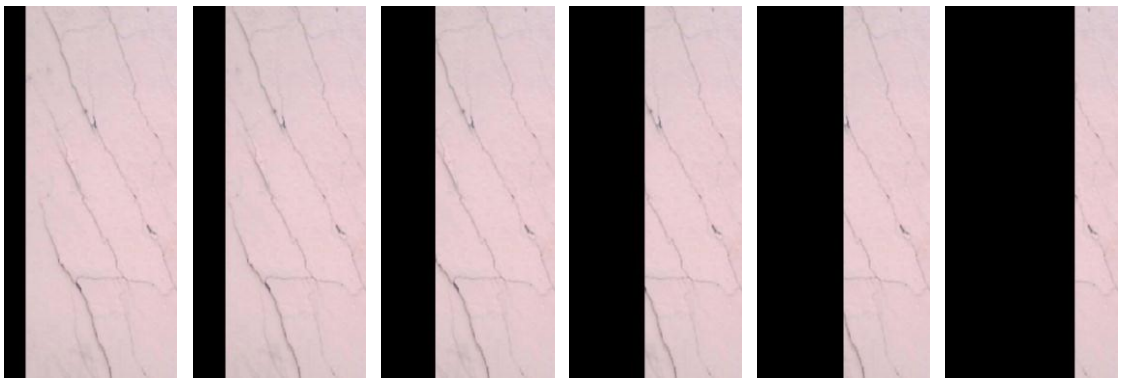


Figure 4.32: Sample results for manufactured horizontal occlusion in a cracked image: (a) 10%; (b) 20%; (c) 30%; (d) 40%; (e) 50%; and (f) 75%

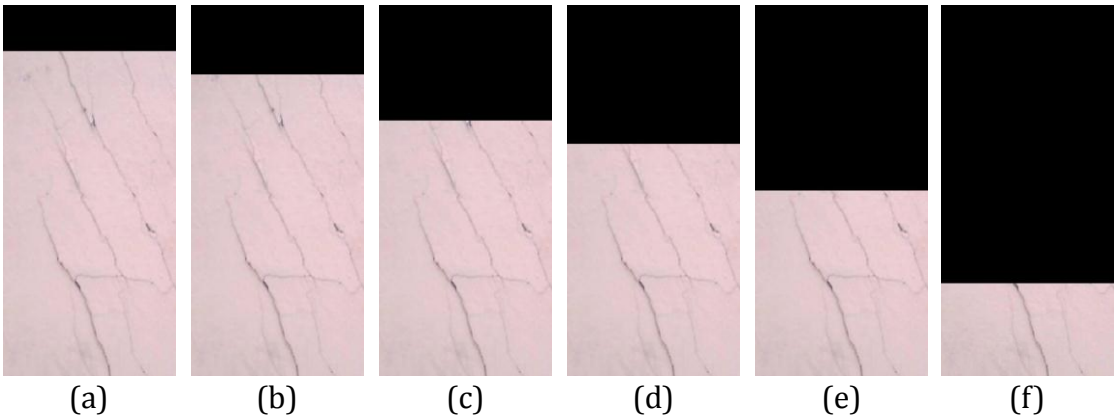


Figure 4.33: Sample results for manufactured vertical occlusion in a cracked image: (a) 10%; (b) 20%; (c) 30%; (d) 40%; (e) 50%; and (f) 75%

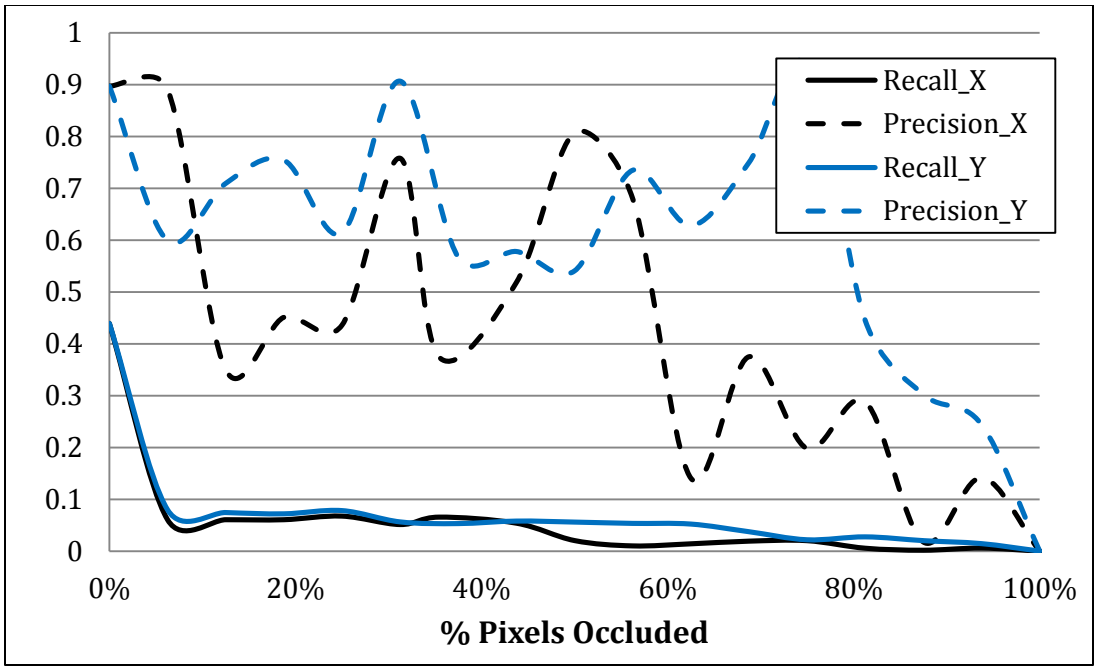


Figure 4.34: Average precision and recall curves for the sensitivity analysis of the automated method in crack property retrieval to occlusion

In looking at these curves, it is evident that the automated crack property retrieval method is very sensitive to occlusion. Although the precision does not immediately fall to zero, the recall does. This is due to the fact that with the portion of the image occluded, it is impossible to see the cracks beyond the occlusion. Thus, any cracks appearing in the region covered by the manipulated occlusion block, are considered false negatives. Ultimately, the precision wavers at each interval of occlusion due to the lack of dependence of the algorithm on how much of a crack is or is not seen. Overall, the sensitivity of both of these methods (spalled and property retrieval) to occlusion will also be highly dependent on the texture and color properties of the occluding object.

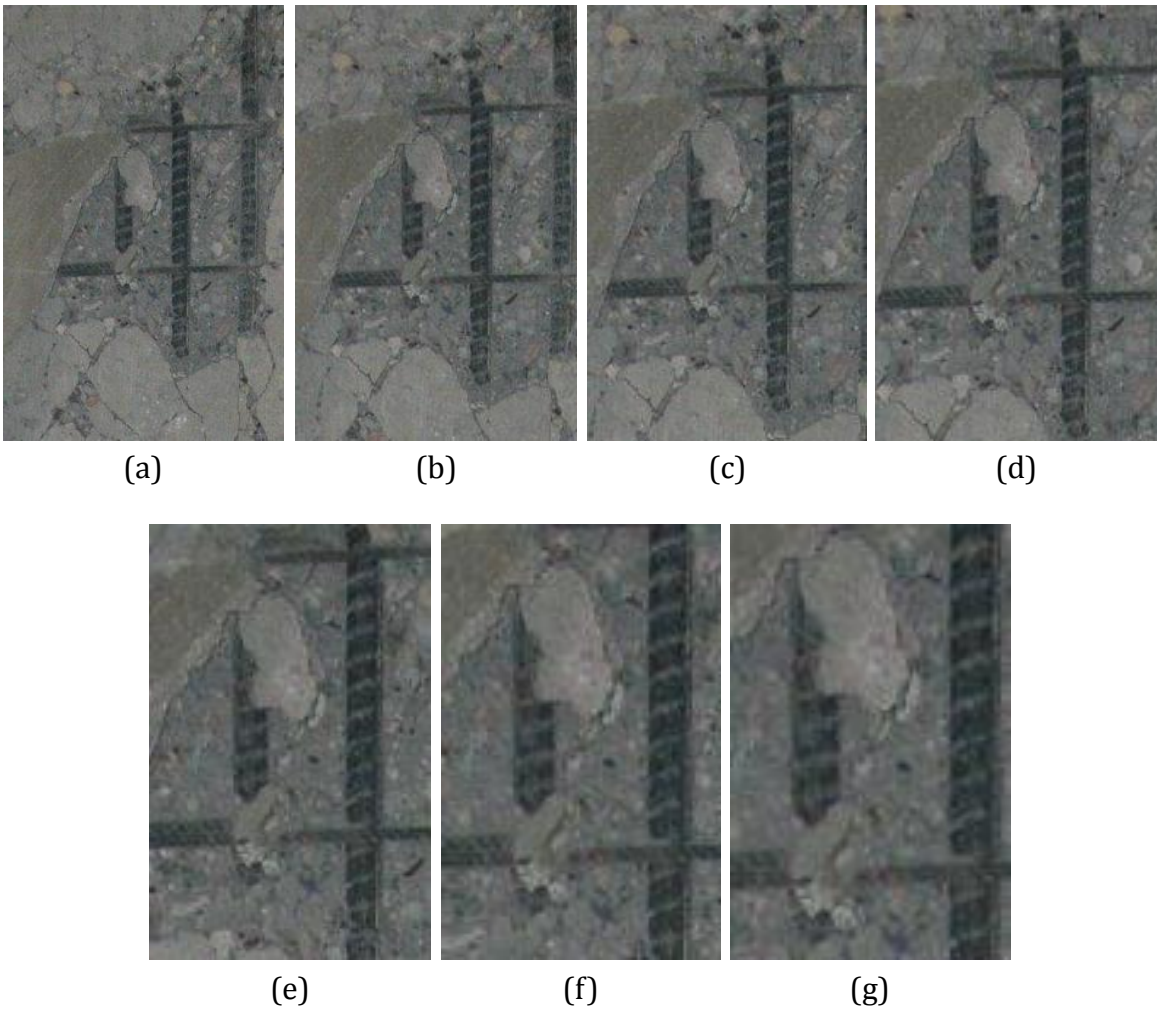


Figure 4.35: Sample spalled image cropped according to zoom ratios: (a) 1.25; (b) 1.5; (c) 1.75; (d) 2.0; (e) 2.5; (f) 3.0; and (g) 3.5

4.6.5. Scale Variation

The final metric considered in the sensitivity analysis of these methods is scale variation. An object's size can vary from one image or video frame to the next due to different amounts of zoom or distances between the observer and the object. In many cases, this is obviously preferred since as the camera lens zooms, small details can be captured. However, the contextual information is lost. In order to test the

sensitivity of these methods to variations in scale, each original image was cropped (from axes with an origin at the center of the original image) according to increasing zoom ratios (1.25, 1.5, 1.75, 2.0, 2.5, 3.0 and 3.5). A representation of the different scaled images is shown in Figure 4.35 for the sample spalled image.

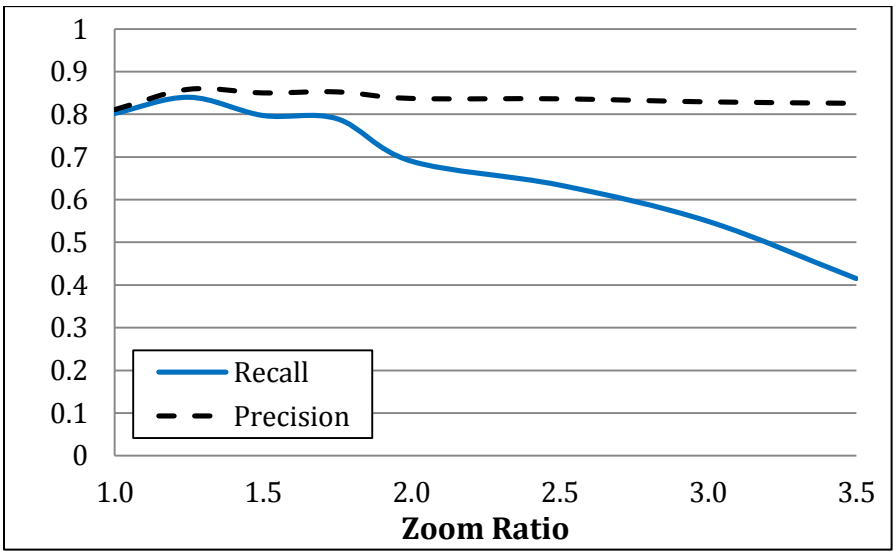


Figure 4.36: Average precision and recall curves for the sensitivity analysis of the automated method in spalled property retrieval to variation in scale

Each image in the spalled image database was cropped in the manner depicted by Figure 4.35 and then the precision and recall were calculated. Then, the average precision and recall for each zoom ratio are shown in Figure 4.36. From this figure, it is evident that contextual information is significant in the spalled property retrieval method. Since the algorithm developed for this method is based on the entropy of the spalled pixels relative to the entropy of the overall image, this is a sensible finding. As the details of the reinforcement and spalled pixels increasingly become the primary and sole focus of the image, the regions without spalling are no

longer visible. This is why the precision value remains consistent as the level of detail is increased: the number of pixels which are not spalled pixels is near-zero. However, within the context of this work, the necessity to view the entire element, and the lack of need to view the damage at a high-zoom, detail level, is an advantage of the method in spalled property retrieval.

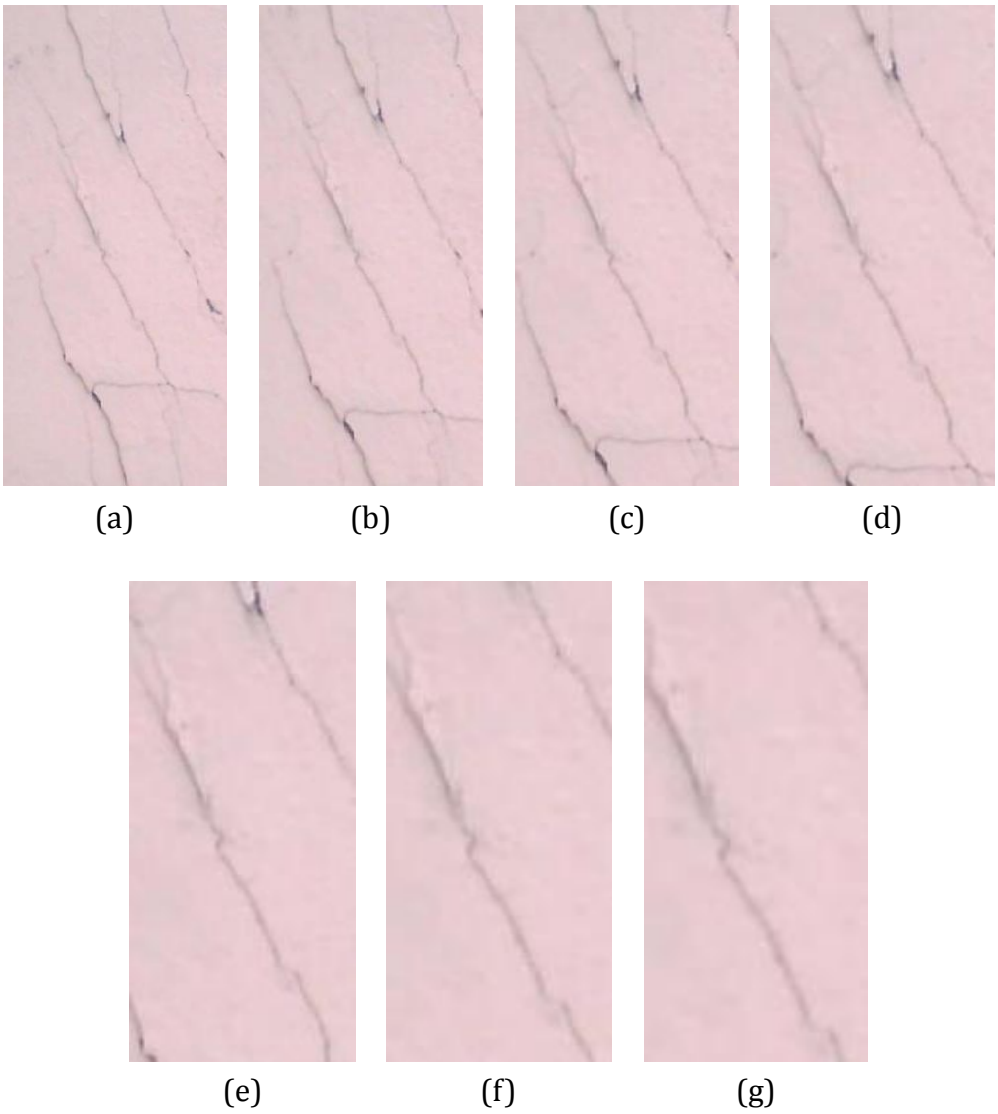


Figure 4.37: Sample cracked image cropped according to zoom ratios: (a) 1.25; (b) 1.5; (c) 1.75; (d) 2.0; (e) 2.5; (f) 3.0; and (g) 3.5

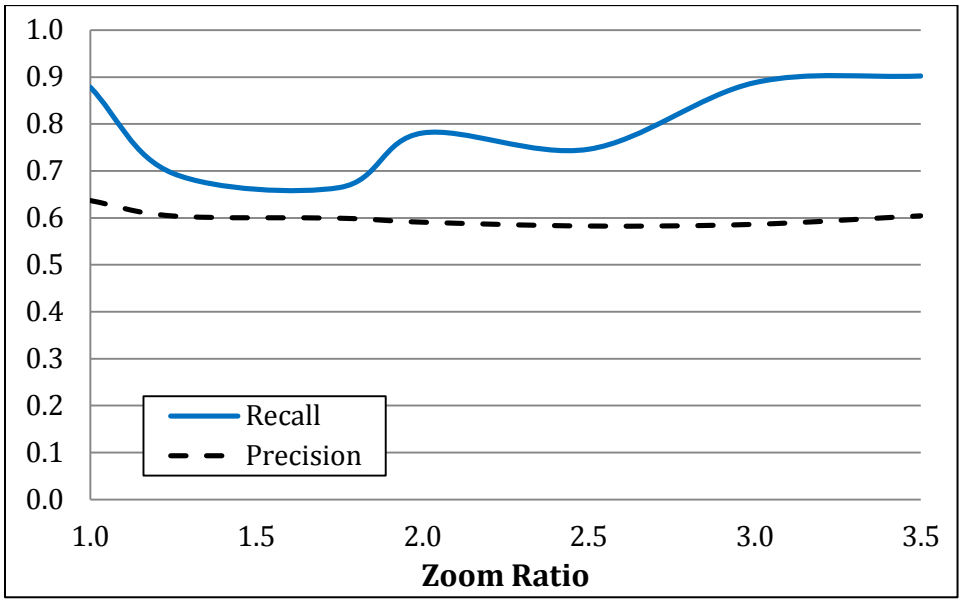


Figure 4.38: Average precision and recall curves for the sensitivity analysis of the method in automated crack property retrieval to variation in scale

The method in automated crack property retrieval is also analyzed with respect to sensitivity to variations in scale. In the same manner, each image in the crack image database is cropped according to the same seven zoom ratios (1.25, 1.5, 1.75, 2.0, 2.5, 3.0 and 3.5) (Figure 4.37). Then the precision and recall for each result is measured and the average precision and recall are depicted by the curves in Figure 4.38. Both of these curves clearly show the scale invariant nature of the method developed in this work for crack property retrieval. The precision values and the recall values alike waver slightly, but in general, the values are consistent. At a high-level of detail and a low-level of detail, the appearance of a crack does not differ. The only differentiation in the image is the number of pixels which are used to represent the crack. As the method employed in this work only considers the

linear nature of cracks and the gradient intensity across the crack pixels, the detection precision and accuracy is not affected by the increased level of detail.

4.7. Closure

This chapter presented two novel methods in automated damage property retrieval for RC column surfaces. The first method, automated spalled property retrieval, is initiated by a local entropy-based threshold characterizing the spalled region of interest on the concrete column surface. Following this, the depth of the spalled region into the column is characterized according to the degree of exposed reinforcement. Thus, any existing exposed reinforcement – transverse and/or longitudinal – is detected. The transverse reinforcement is detected using an adapted percolation-based region growing process, and the longitudinal reinforcement is detected using the EmguCV template-based matching algorithm. Each of these algorithms are performed in the CMYK color space as this space was determined to be the dominant space for reinforcement regions amidst spalled concrete surfaces. In addition to the depth of the spalled region, the length of the spalled region along the longitudinal axis of the member is also detected by way of a novel connected component algorithm and distance transforms.

The second method, automated crack property retrieval, is intended to provide the evaluator with the dominant crack pattern on the concrete surface. In this method, initially the crack points and locations of crack points on the RC element surface are detected by way of an adapted percolation-based region growing procedure (very similar to that used in the detection of transverse

reinforcement). Then, by using a Euclidean distance transform and a binary image thinning algorithm, the width, length and orientation of individual crack segments are measured, as well as the spacing between segments of similar orientation. Expending all of this information, the surface crack pattern is classified as one of the following: shear, vertical, horizontal or a combination of any of these two types.

The properties retrieved in each of these methods are related to the detected dimensions of the column surface in order to convey relative and meaningful measurements regarding the extent of damage to the column surface. The methods were each implemented into a prototype developed in Microsoft Visual Studio .NET. A database of approximately 200 different images of damaged and non-damaged RC structural elements was created. The database included images from damaged structures in Haiti following the earthquake in January 2010, images from the NEES earthquake database and images from non-damaged buildings and bridges in the Atlanta area. This database was employed in order to test the performance of each of these methods in damage property retrieval. It is evident based on the average measurement error for the spalled length measurements (L_S/b – 4.22% and L_T/b – 8.05%) and for the crack property measurements (θ_C – 3.29°, L_C/b – 2.21%, W_C/b - 0.35% and S_C/b – 6.06%), that the methods presented for automated spalled property retrieval and crack property retrieval are more than adequate. Further, the average accuracy for the methods in spalled depth classification (87.53%) and crack pattern classification (94.28%) reinforce the successful nature of the methods presented here in automated damage property retrieval.

In addition, both of these methods were analyzed with respect to their respective sensitivity to five frequent intrinsic (camera parameters) and extrinsic (environmentally- or user-dependent) variables: (1) illumination; (2) blurring; (3) camera-shift; (4) occlusion; and (5) scale variation. Based on these analyses, the benefits and limitations of these two methods in general, and with respect to the application outlined for this research specifically, were illustrated.

CHAPTER 5

DAMAGE INDEX ESTIMATION

In order for the automated computer vision detection and property retrieval methods discussed in Chapter 4 to be meaningful in the field of structural engineering, and in specific, in the application of post-earthquake safety and structural assessments, it is necessary to find a standard way to evaluate RC columns based only on this visible damage which is observed after the earthquake has occurred (at the residual state). Typically, when subjected to earthquake loading, the damage pattern, response mechanism and failure mode of RC columns vary based on geometry, material properties and design details. Therefore, the observed (residual) damage must be accounted for regardless of the geometry, material properties and design details for any specific column, and it should be correlated to the maximum drift which the column was exposed to during the earthquake. In this chapter, first the link between column maximum drift capacity and the observed visual damage at that maximum capacity is defined. The damage progressions for two critical response mechanisms were discussed in Section 3.4, and a set of specific damage state indices was introduced (Bearman, 2012). As mentioned previously, these indices relate the damage observed at different maximum column states to maximum drift capacities. The distinction between the damage detected in the context of this research (residual) and that observed at maximum drift capacity is then presented, and a conservative set of damage state indices are introduced which relate the residual observed damage to the maximum

drift capacity of the column which can ultimately be used to assess the vulnerability of the entire building.

5.1. RC Column Assessment Variables

There are several variables which influence the damage state of RC columns when subjected to a seismic event, such as material properties, geometry, design details, loading variations, failure modes, etc. Due to the nature of this research, the damage state is founded entirely on the visible observed damage on the column surface. The column's axial load can be easily deduced from the general building geometry, and the response mechanism should also be defined by the observed visible damage. In this section, these two variables are discussed further with respect to how they affect the progression of damage on RC columns subjected to earthquake loading.

5.1.1. Response Mechanism

Three response mechanisms are pertinent for RC columns subjected to axial load and cyclic lateral (earthquake) loads: (1) flexure-critical; (2) shear-critical; and (3) flexure-shear critical. These response mechanisms are defined with respect to the most likely failure mode of the column. The focus of these research efforts is limited to flexure- and shear-critical column responses due to the work which has been performed regarding the correlation between the progression of visible damage and maximum drift capacities for these failure modes (Bearman, 2012). The details of these two response mechanisms were discussed in Section 3.4. A flexure-shear response is essentially a combination of a flexure response and a shear response.

The column behaves similar to that of a flexure-critical column through the cracking to yielding stages at which point the dominating mechanism will shift to shear.

5.1.2. Axial Load

As seen in Section 3.4, the axial load on an RC column does not affect the progression of damage in flexure- or shear-critical columns; however, it does influence the displacement ductility or drift capacity (Bae, 2005). The higher the axial load, the more rapid the damage will progress and the lower the drift demand on the column. Because drift is the engineering demand parameter (EDP) chosen for the purpose of this research, the magnitude of the axial load should also be taken into consideration. Thus, the damage progressions which are presented in Section 3.4 for the two response mechanisms mentioned are identical regardless of the axial load. However, the associated drift capacity with each stage of the damage progression is specific to the axial load on the column. In each stage, a drift capacity value for a column with a low axial load is specified as well as a drift capacity value for a column with a high axial load. The critical axial stress is identified as $0.5f'_cA_g$ such that any axial load equal to, or above this value is considered high (HAL), and any value below this value is value is considered a low axial load (LAL). This distinction is also considered in the development of damage indices related to the residual observed damage state for RC columns.

5.2. Linking Maximum Damage with Performance for RC Columns

The post-earthquake vulnerability of the structure to collapse can be estimated using individual column damage and response-mechanism information. The results

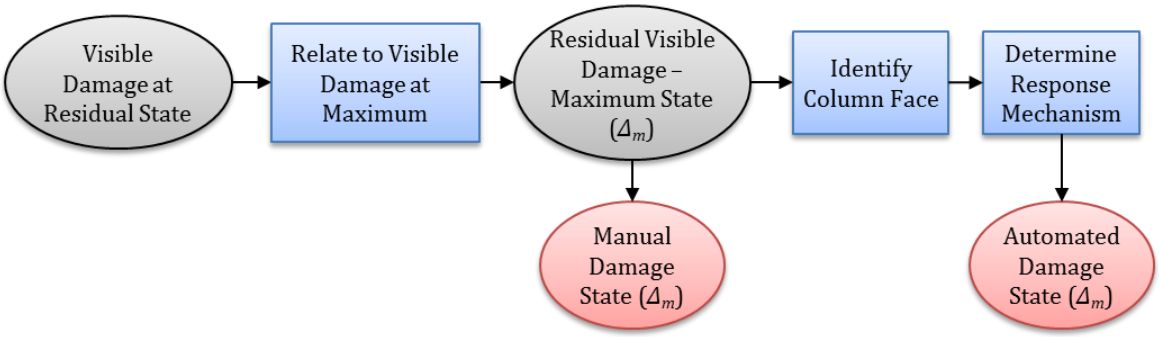
of previous research studies provide a basis for establishing links between damage patterns and response mechanisms for RC frame members. In order to know the characteristics of the damage patterns associated with specific response mechanisms and how these damage patterns evolve due to earthquake loading, sufficient data in the on-going tests have been used to classify frame member response on the basis of observed damage patterns. From existing column experimental work, this relationship between the maximum drift capacity and the visible damage associated with this capacity has been established (Bearman, 2012).

As has already been mentioned, the purpose of this research is to determine the existing state and/or safety of the structure in a post-earthquake scenario. Therefore, in order to determine damage states and corresponding maximum drift capacities of RC columns based on visual cues, a database of RC column tests (Sezen, 2002; Bae, 2005) and test images (Eberhard et al., 2010; Sedra et al., 2010; PEER, 2011) have been studied. Knowing a well-defined range of drifts at which a damage state occurs and identifying the failure mechanism based on images from past research can be used to provide a more consistent and generalized understanding of the behavior of structural elements. The damage state tables displayed in Section 3.4 (Table 3.3 and Table 3.4) demonstrate these drifts and observed damage progressions for each response mode.

5.3. Linking Residual Damage with Performance for RC Columns

Although the results summarized in Table 3.3 and Table 3.4 provide a comprehensive description of the relationship between maximum drift capacity and the visual damage associated with this capacity for RC columns, a correlation still

does not exist between the residual condition (visual damage cues) of the column and the column's maximum drift capacity. This involves three further steps before the final drift-damage correlation can be determined (Figure 5.1): (1) distinguishing between residual and maximum damage; (2) defining a relationship between the visible damage and the specific concrete surface; and (3) defining a relationship between the visible damage and the response mechanism. These three steps will be discussed in the remainder of this section.



*Δ_m = maximum drift capacity of RC column

Figure 5.1: Overview of method in automated damage index estimation based entirely on damage observed at the residual state for RC columns

5.3.1. Residual Drift vs. Maximum Drift

As mentioned, with respect to the visual damage detection stage of this work, the damage detected is based on the column in its residual state. In order to provide a reliable estimate of the column's performance based on the results from the automated methods in crack and spalling property retrieval presented in Chapter 4, the damage state definitions should be broadened such that the maximum drift

capacity for a given damage state index corresponds to the observed visible damage to the column in its residual state. Therefore, the damage observed at the residual and maximum drift capacities must be compared, and any distinction between the two should be represented in the final damage state definitions provided at the end of this section.

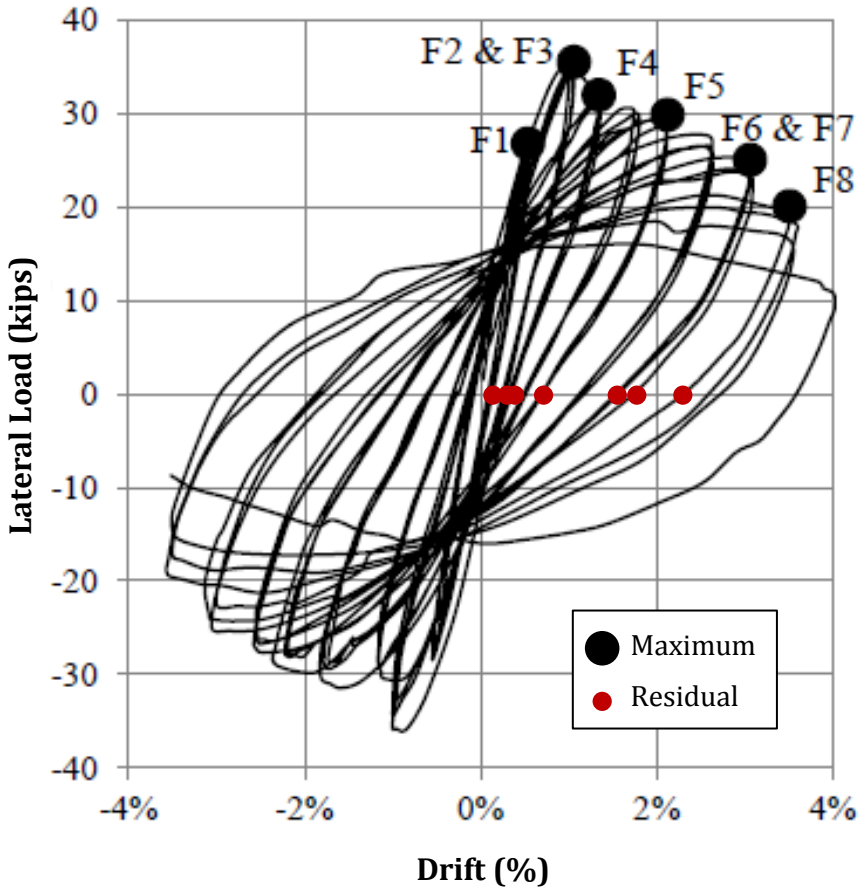


Figure 5.2: Example of lateral load vs. drift history for flexure-critical column with residual drift locations marked by red points

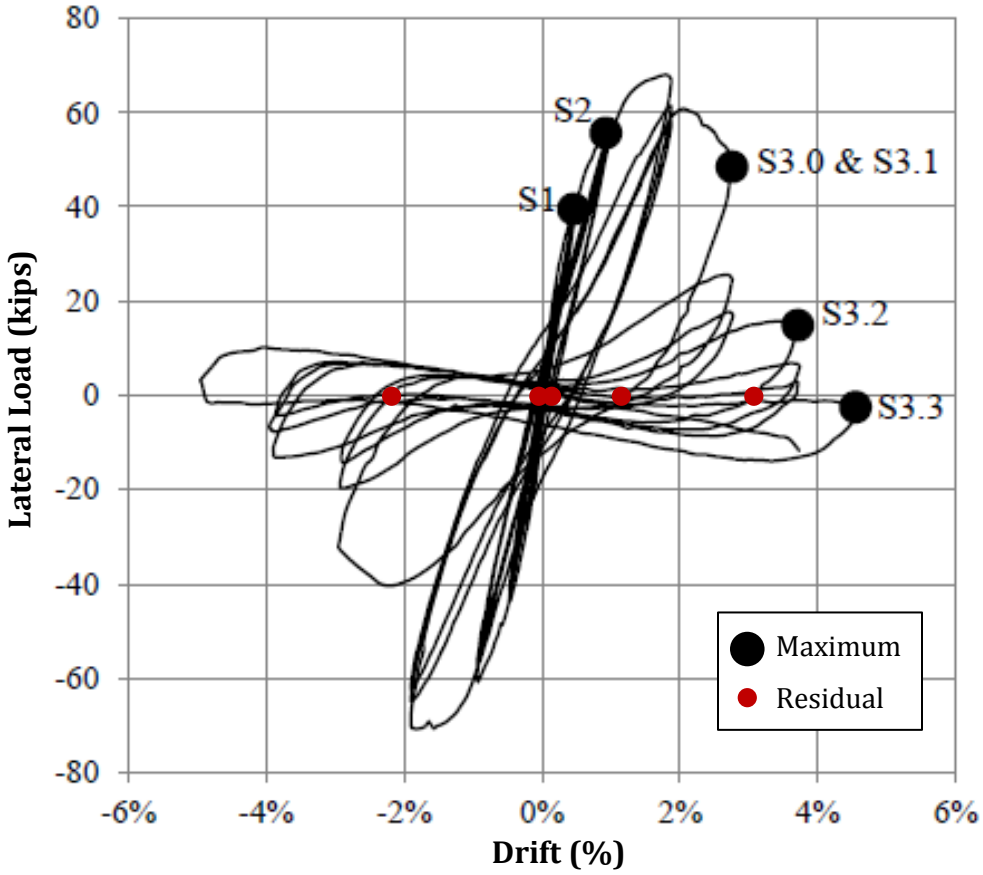
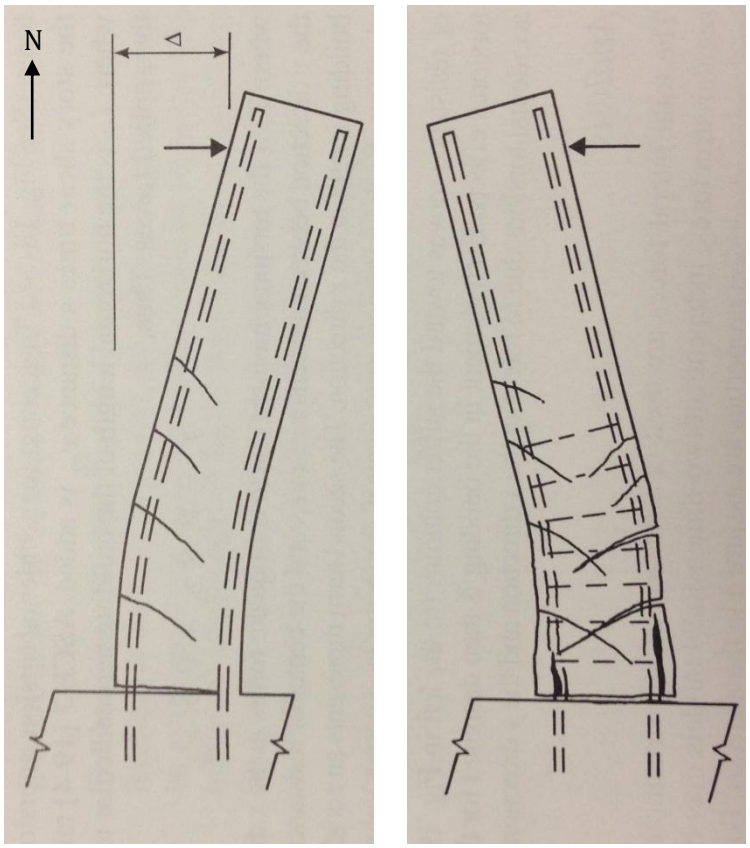


Figure 5.3: Example of lateral load vs. drift history for shear-critical column with residual drift locations marked by red points

In Section 3.4, the maximum drift capacities which corresponded to each damage state definition (marked by black dots in Figure 5.2 and Figure 5.3) were discussed. In this section, the damage observed at residual drift capacities corresponding to each damage state definition is now discussed. These points on the lateral load vs. drift histories are marked by red dots in Figure 5.2 and Figure 5.3, and correspond to the column's resting place when returning from the maximum experienced drift. There is no increase of lateral load between these two states, and thus, the damage does not propagate any further until the load begins to increase

again. If the damage were to propagate, this would provide a more conservative estimate for the column's condition, and so this is not a concern. As discussed in Section 3.4, when an RC member is subjected to cyclic loading, cracks begin to form (Figure 5.4 (a)). When the load reverses, these cracks close and new cracks form (Figure 5.4(b)) (Wight and MacGregor, 2009). Thus, this creates the distinction between the appearance of cracking at the residual and maximum state of the column.



(a) (b)

Figure 5.4: Reinforced concrete column subjected to cyclic loading: (a) cracks due to load in the East direction; and (b) cracks due to load in the West direction (Wight and MacGregor, 2009)

Table 5.1: Observed state of damage at maximum and residual column capacity

Damage Property	Maximum	Residual
Crack width	W_C / b	May have closed
Crack Length	L_C / b	No change unless covered by spalling
Crack Orientation	Θ_C (angle WRT x-axis)	No change unless covered by spalling
Crack Spacing	S_C / b	No change unless covered by spalling
Spalled Length	L_S / b	No change
Spalled Length	L_T / b	No change
Spalled Depth	Extent of exposed rebar	No change

*where *b* = width of the column

Based on the description of the damage progressions for both flexure- and shear-critical columns provided in Section 3.4, the variations in damage appearances at maximum and residual drift can be deduced (Table 5.1). The third column in Table 5.1 specifies that in all cases, spalled regions will not be noticeably different from the maximum to residual state. Once the concrete has spalled off, it will remain that way, and the reversing of the lateral load will not cause anything more than flaking of concrete which has in fact already lost contact with the core, and so, the visual cues associated with spalling will not differ from the maximum to the residual state (Bearman, 2012). Alternatively, with respect to cracking, there is some variation from one state to the other. The crack length, spacing and orientation (since measured with respect to the column's transverse axis) will not

technically vary in appearance from the maximum to residual states—and this is true in general (Figure 5.4). However, in some cases, such as with longitudinal cracking on the side faces for shear-critical columns (S3.1) or longitudinal cracking on the flexural faces for flexure-critical columns, the cracks may not be apparent any longer in the residual condition of the member. In these cases, since spalling follows quickly after cracking in the damage progression, the cracked regions may be entirely covered by the spalled region (Bearman, 2012). Thus, in these types of situations, the column should be evaluated according to the most extreme type of damage which can be observed rather than whether or not all types of damage leading up to that exist. The only damage property which may exhibit an intrinsic variation from the maximum displacement of the column to the residual state is crack width. As the column drifts back, the cracks which have opened due to stresses acting in one direction may be closed. As a result, in general, crack widths will not be considered for the classification procedures in the automated evaluation method. Eventually however, the lateral load demand will increase such that the cracks remain open. These crack widths can be accounted for with certainty in the automated method.

5.3.2. Column Surface Characterization

The output of the automated method in concrete column detection which was presented in Section 4.1 is an individual concrete column surface. However, in the damage progressions displayed in Section 3.4, both the flexural and side faces of the column are considered in order to accurately determine the damage state in which the column currently belongs. At this time, in order to fully automate the procedure

in RC column damage index estimation based on the information which can be retrieved employing the methods presented in Chapter 4, it is necessary to first determine which face of the column has been detected (direction of applied cyclic lateral load).

At the different stages in the damage progression for flexure- and shear-critical columns, the two faces experience different types and degrees of damage. At the final damage state, a column of either type will likely have some visible damage of every type on both faces. Therefore, the dominant damage types for each column face, regardless of the stage in the damage progression or the response mechanism, are considered. Since this distinction must be made for every column surface, it is ideal to find a damage characteristic included in one of the initial stages of the damage progressions which can distinguish the two column faces from one another. In general, flexural cracks, which initiate on the flexural column faces in the first stage of the damage progression for both flexure- and shear-critical columns, will not propagate on the side faces at later stages. Therefore, the existence of flexural cracks on the detected surface serves as the first classifier for the method in automated damage state estimation. This approach will err on the side of over-classifying column faces as "flexural," and so the overall procedure for automatically estimating the damage state of a column surface classified as "flexural" will take this into account by providing a sufficiently conservative evaluation.

5.3.3. Response Mechanism Characterization

Although the damage progression and visible observed damage has been discussed thus far with respect to the response mechanism or failure mode of the RC column,

this entity has yet to be determined from an automated perspective. Although the distinction between the two response mechanisms is evident when considering the manner in which the damage progresses, without this sort of "contextual" information it is more complicated to ascertain exactly which type of response the column is experiencing. For example, the knowledge of how and when a crack propagates is not available, but only the fact that a crack of a certain relative length, orientation and width is present. Additionally, as mentioned previously, the progression of spalling which often follows quickly after cracking causes these two stages to be indistinguishable from one another in those cases. However, the response mechanism must be defined so that the corresponding damage state also can be defined. The visual cues which are specific to the two distinct damage progressions (flexure-critical columns (Table 3.3) and shear-critical columns (Table 3.4)) are assessed. A summary of the dissimilarities between the visual damage cues observed for each of the response mechanisms is provided in Table 5.2. Also in this table, the capability of the method described in this work to detect the specific distinguishing characteristic and the hypothesized reliability of using the specific characteristic are provided.

The first two potential distinguishing characteristics concern the location and shape of spalling on the column. In the damage progression for flexure-critical columns, the spalled region is specified as prevailing only in the top and bottom $1/4^{th}$ (flexural face) or $1/5^{th}$ (side face) of the column (Table 3.3). However, in the damage progression for a shear-critical column (Table 3.4), if spalling does exist, it can be found at any location along the height of the column. This means that if a

spalled region is detected in the middle section (½ (flexural faces) or 3/5ths (side faces)), the column response mechanism can be definitively classified as shear. The alternative though, is inconclusive: if spalled regions are only detected at the top and bottom portions of the column, the response mechanism cannot be defined as flexure with as high a level of confidence. The shape of the spalled region will also differ depending on the response mechanism. However, since the method in spalled region detection is based on the inherently indistinct shape, any shape or edge information would be fairly unreliable. Since the methods created and presented in this work are capable of determining the location of the spalled regions on the surface, this will be employed as a partial characterization metric in determining the column response mechanism.

The next three distinguishing damage characteristics reference any existing shear and longitudinal cracks on the observed column face. The location and the width of shear cracks, as well as the length of longitudinal cracks, can be distinguishing parameters depending on the circumstances (i.e. state in damage progression). The shear crack location and the longitudinal crack length can be employed as response mechanism classifiers with the same amount of confidence as the location of spalled regions. Shear cracking occurs only at the top and bottom $1/3^{rd}$ of flexure-critical columns, but can occur at any height on a shear-critical column. In addition, longitudinal cracks typically do not extend further than $1/3^{rd}$ the length of the column from the joint in flexure-critical columns but may propagate all along the length of shear-critical columns. Hence, if shear cracks are found in the middle $1/3^{rd}$ of the column or longitudinal cracks are found which

extend into this zone, the column can definitively be specified as shear-critical. On the contrary, a shear crack's existence or longitudinal crack length contained in the top or bottom 1/3rd of the column could indicate either response mechanism. Each of these visual cues will be utilized in the overall classification of the column response mechanism.

Once a shear-critical column has sustained a sufficient amount of damage, it has been stated that the shear cracks will maintain a certain width, no longer closing upon load reversal. Thus, if cracks are observed (anywhere) on the surface which is beyond this specified threshold, the column is conclusively shear-critical. The lack of shear crack widths exceeding this value does not indicate a flexural response mechanism in any case. The instances where the crack width exceeds the specified value will be utilized to help classify the response mechanism of the concrete column surface.

The buckling length of the longitudinal reinforcement is the last distinguishing damage characteristic found in the typical damage progressions for flexure-critical and shear-critical RC columns. Although the actual appearance of a buckled longitudinal bar is consistent and not dependent on the response mechanism, in flexure-critical columns, the buckling length is typically shorter than in shear-critical columns. As mentioned in Section 3.4, longitudinal bar buckling in flexure-critical columns may occur between transverse reinforcing ties or across several due to the tight spacing of ties in columns of this type. Thus, in order to attempt to automatically define the response mechanism, the method will create a metric which stems from this characteristic difference of bar buckling.

Table 5.2: Distinction between visual cues in damage descriptions for response mechanism characterization

	Flexure-Critical	Shear-Critical	Method Capability	Reliability
Spalled Location	Only at top and bottom 1/4th (flexural) or 1/5th (side)	Any height	Yes	Medium
Spalled Shape	Indistinct	Triangle or parallelogram	No	Medium
Shear Crack Location	Top and bottom 1/3rd	Any height	Yes	Medium
Shear Crack Width	Indistinct	Residual ≥ 0.3in. (HAL), 0.5 in. (LAL)	Yes	Medium
Longitudinal Crack Length	No more than 1/3rd of the column height	May run the entire height of column	Yes	Low
Buckling Length	Shorter	Longer	Yes	Low

*where *HAL* = high axial load; *LAL* = low axial load

5.3.4. Damage State Definitions

Now that the relationship between the damage observed at the maximum drift and that observed at the residual drift is determined and the relationships between the detected column surface and response mechanism and the visible observed damage have been explored, a set of damage state definitions, similar to those defined in Table 3.3 and Table 3.4, but which relate only the visible damage at the residual state to the maximum drift capacity of the column should be developed. In addition, as mentioned previously, the column response mechanism should also be determined automatically. The visible damage which helps to distinguish these two

mechanisms (Table 5.2, Section 5.3.3) is considered and portrayed in this set of damage state definitions. However, the impact of these characterizations as well as that of the "face" of the column which is being assessed is illustrated more substantially in the Implementation section. The resulting damage state definitions (Table 5.5) take into account each of the retrieved properties discussed in Chapter 4.

The first three damage states listed in Table 5.3 (D0-D2) represent the damage states necessary for those situations where a columns damage response mechanism cannot be determined. When any one of these damage states is specified, conservative values of drift which correspond to the flexure- and shear-critical equivalents are used. In specific, the first state (D0) is a descriptor for any column surface exhibiting no damage. The next state (D1) combines the flexure-critical and shear-critical damage state definitions regarding flexural and longitudinal cracks on the flexural column face (F1, F2 and S1). Since, in each response, the damage progression is identical when these cracks are evident, the response mechanism cannot be determined if this is the only damage which is observed at the residual column state. Thus, the highest of the three sets of drift values shown in Tables 3.3 and 3.4 are associated with this state in Table 5.3. The final of these three damage states (D2) illustrates the uncertainty of the condition of a column with shear cracks only on the top and bottom 1/3rds of the surface. This state encompasses both F3 and S2 from the flexure-critical and shear-critical damage progression tables, respectively. Thus, the drift values for F3 from Table 3.3 controls and is specified here.

The second set of damage indices in Table 5.3 represent those shear-critical indices (from Table 3.4) which can also be defined with confidence by way of the automated process based entirely on the visual cues presented here. The first two shear-critical damage indices consider the location and width of shear cracks as discussed in earlier sections. In some cases, the column will be designated as D2, in other cases, S2 or S3.0, and the associated, more accurate drift values, may be able to be specified. As the states progress into representations of columns which have undergone more significant damage, the distinction between response mechanisms is always possible. This is because there are more available visual damage descriptors. Thus, the final three shear-critical damage states in Table 5.3 are nearly-identical to those listed in Table 3.4: (1) longitudinal cracking propagating anywhere along the side faces(S3.1); (2) concrete spalling all along the side faces (S3.2); and (3) longitudinal bar buckling and core concrete crushing (S3.3: S3.3 and S3.4 from Table 3.4). Since, in the shear-critical response, a column will most likely exhibit all of this damage concurrently, these damage descriptions are used in conjunction with one another to establish the existing state of the column. The damage state associated with the most progressed index is designated as the current state of the column.

Table 5.3: Summary of vision-based damage indices for RC columns

Damage State	Damage Description	Response Mechanism	Drift (%)	
			LAL	HAL
D0	• No damage	Unknown	0.0	0.0

D1	• Flexural cracks:
	  • Top and bottom 1/3rd of column
	  • Span width of column
	• Longitudinal cracks:
	  • Top and bottom 1/3 of column
		Unknown	1.0	0.75

D2	• Shear cracks occurring at the top and bottom 1/3rd of column	Unknown	2.0	1.75
S2	• Shear cracks occurring in the middle 1/3rd of the column	Shear	0.5	0.5
S3.0	• Widening and localization of shear cracks • W_c / b ≥ 1/60	Shear	2.0	1.75
S3.1	• Longitudinal cracking on side faces	Shear	2.5	1.75
S3.2	• Concrete spalling on side faces	Shear	2.5	1.75
S3.3	• Longitudinal bar buckling/crushing of core concrete • L_T / b ≥ 1	Shear	2.5	1.75
F4	• Concrete spalling • Top and bottom 1/4th or 1/5th of respective faces	Flexure	1.5	0.75
F5	• Concrete spalling exposing longitudinal steel	Flexure	2.0	1.0
F6	• Longitudinal bar buckling /crushing of core concrete • L_T / b ≥ 1/2	Flexure	6.0	3.5

*where *b* = width of the column; *LAL* = low axial load; *HAL* = high axial load; L_T = distance between extreme exposed transverse reinforcement bars

The last three indices listed in Table 5.3 (F4, F5 and F6) refer to the flexure-critical damage states (from Table 3.3) which the automated method is capable of estimating with surety. When the column is damaged such that spalling is apparent,

as seen in the shear-critical damage states which are computable, the flexure-critical damage states beyond this type of damage are also computable. Therefore, based on the capabilities of the methods in automated damage property retrieval, the damage index can be estimated for the following: (1) concrete spalling at the top and bottom column sections (F4); (2) concrete spalling exposing longitudinal reinforcement (F5); and longitudinal bar buckling/crushing of core concrete (F6: F6 and F7 from Table 3.3).

In general, the automated method in damage index estimation is capable of determining the existing state of a flexure-critical or shear-critical column to the extent of bar buckling and crushing of core concrete. In an RC column (flexure- and shear-critical), since longitudinal bar buckling and core crushing occur simultaneously, the drift values are identical for these two states and thus, without visibly identifying the existence of core crushing, it can be considered a capability of the automated method based entirely on the existence of bar buckling. However, beyond concrete core crushing in a flexure-critical column, based on the damage properties retrieved by way of the methods presented in Chapter 4, the damage index cannot be estimated. However, the drifts associated with the most extreme progression of damage are employed for the final flexure-critical damage index in order to account for this level of uncertainty.

5.3.5. Automated Damage Index Model Design

Now that the comprehensive damage state indices have been defined based entirely on the visible damage properties retrieved using the automated methods presented in Chapter 4, it is necessary to create the physical link which connects these pieces.

The model in automated damage index estimation is designed as two separate processes: one for the case where the face which is detected is the flexural face, and one for the case where the face which is detected is the side face. A flowchart illustrating each of these processes for determining the response mechanism, damage state and drift values for a given column surface are displayed in Figure 5.5 (flexural face) and Figure 5.6 (side face). With these processes, the response mechanism, damage index and associated EDP (maximum drift capacity) for a given column surface can be automatically determined.

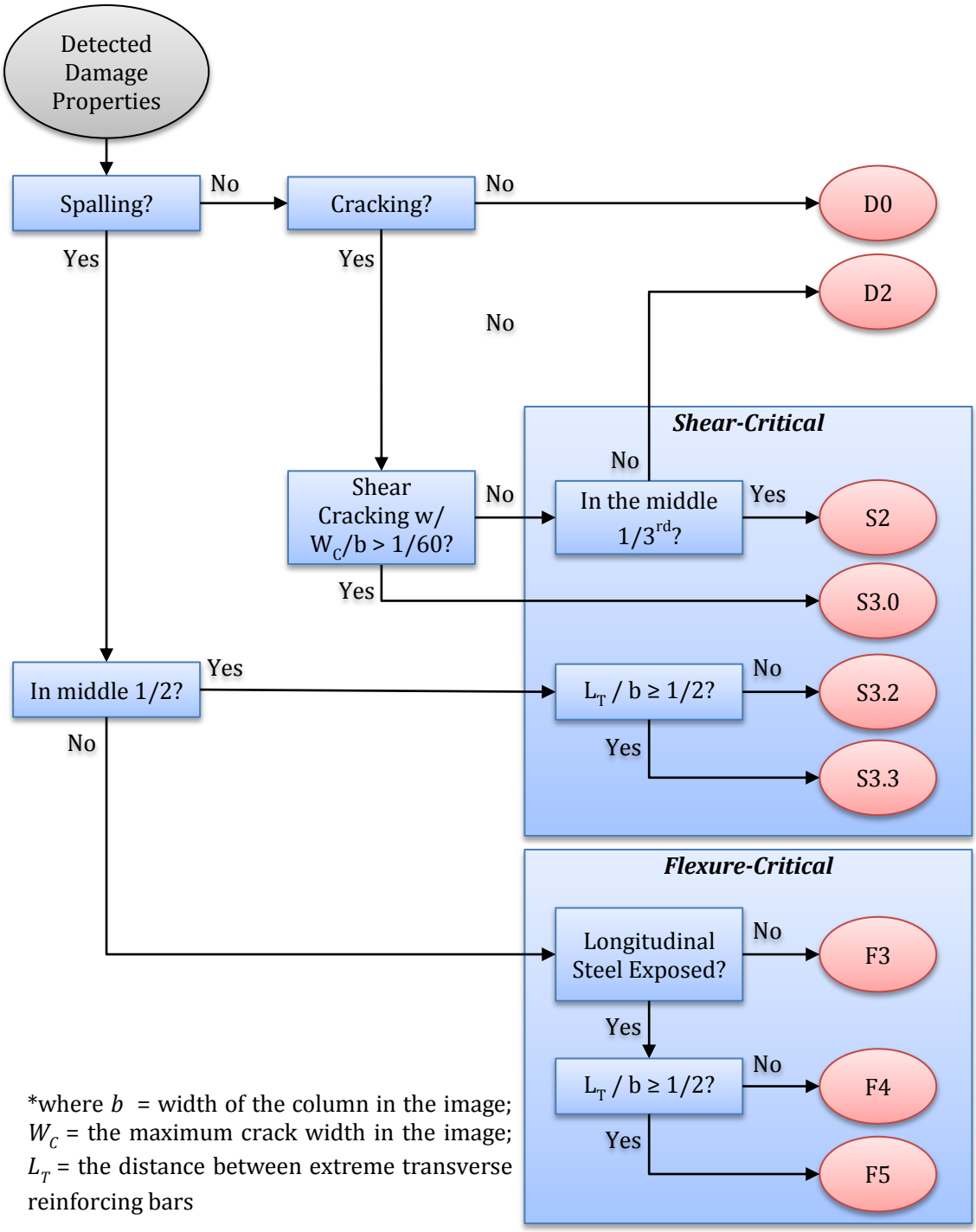


Figure 5.5: Flowchart for automated damage index estimation when surface detected is the flexural face

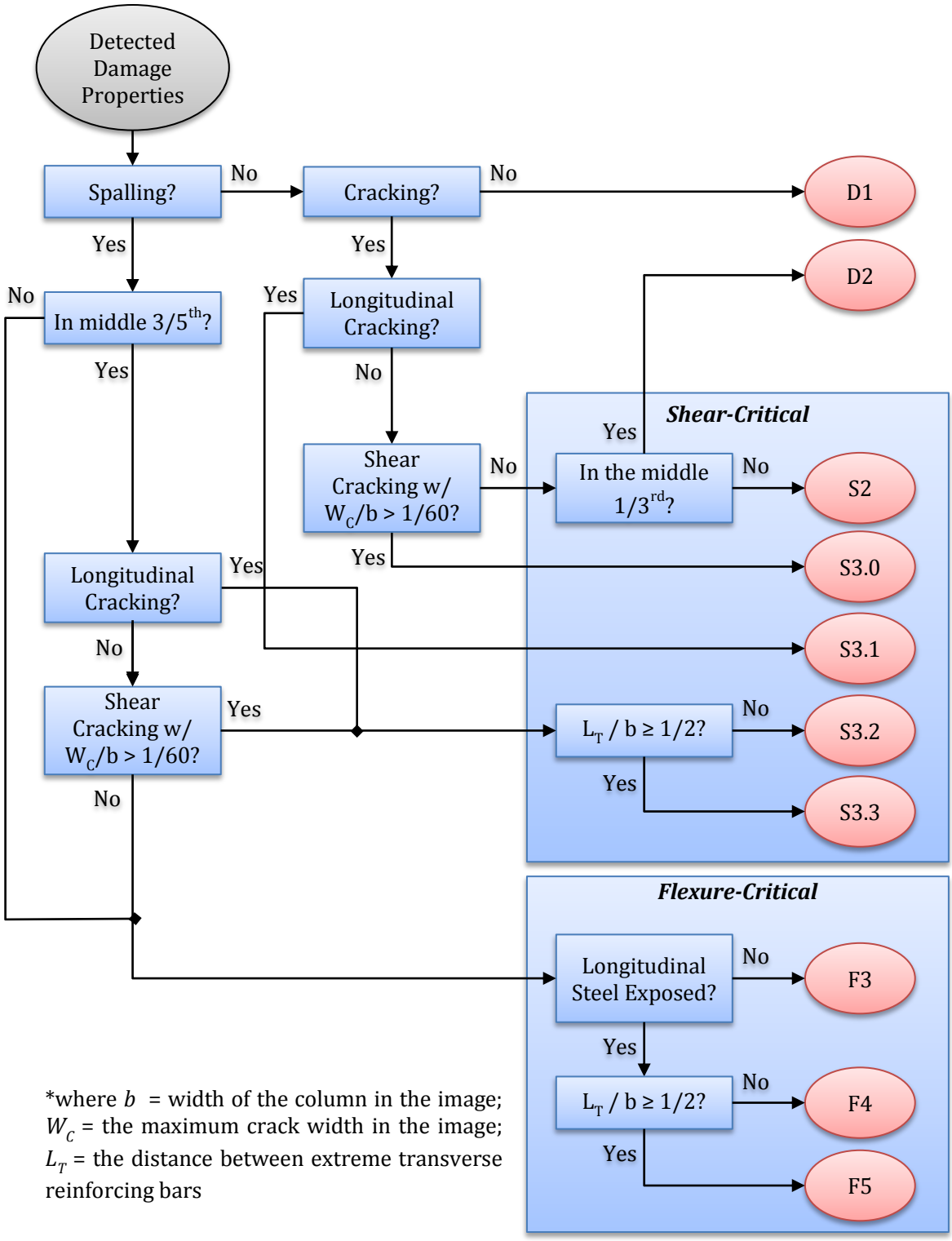


Figure 5.6: Flowchart for automated damage index estimation when surface detected is the side face

5.4. Implementation and Results

In this section, the implementation of the model used to translate the automatically detected RC column damage properties into the damage state indices and associated maximum drift capacities is presented. The results for the verification of the database's accuracy are also presented and an example is discussed.

5.4.1. Implementation of Automated Damage Index Model

A prototype of the model for automated damage index estimation (Table 5.3 and Figures 5.5 and 5.6) was developed using Microsoft Visual .NET, integrally with the methods in automated damage detection and property retrieval presented in Chapter 4. The model was also implemented and integrated into the larger prototype developed by the Construction Information Technology Laboratory, formerly at the Georgia Institute of Technology, as an independent module.

5.4.2. Automated Damage Index Model Performance

For the method in automated damage index estimation, a database of 50 damaged RC column images was used. The appropriate damage index for the column was provided manually, and the results from the automated method in damage index estimation were compared with these manual indices. The performance of the method in automated damage index estimation (based on the automatically retrieved damage properties discussed in Chapter 4) for the entire classification procedure, as well as for each individual class, is calculated using Equation 5.1 (Equation 4.12 from Chapter 4).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad \text{(Eq. 5.1)}$$

Table 5.4: Results for automated method in damage index estimation

		Manually Classified										
		D0	**D1**	**D2**	**S2**	**S3.0**	**S3.1**	**S3.2**	**S3.3**	**F4**	**F5**	**F6**
Automatically Classified	**D0**	0	0	0	0	0	0	0	0	0	0	0
	D1	0	0	0	0	0	0	0	0	0	0	0
	D2	0	0	0	0	0	0	0	0	0	0	0
	S2	0	0	0	0	0	0	0	0	0	0	0
	S3.0	0	0	0	0	0	0	0	0	0	0	0
	S3.1	0	0	0	0	0	0	0	0	0	0	0
	S3.2	0	0	0	2	2	1	15	8	4	0	2
	S3.3	0	1	0	1	1	2	2	1	0	2	0
	F4	0	0	0	0	0	0	0	0	1	0	0
	F5	0	0	0	0	0	0	1	0	0	0	0
	F6	0	0	0	0	0	0	0	0	0	0	0
Accuracy		**1**	**0.96**	**0.98**	**0.94**	**0.92**	**0.94**	**0.52**	**0.66**	**0.92**	**0.94**	**0.96**

For the 50 test images in the damaged image database, the overall accuracy for the classification of the damage index of RC columns was calculated as 88.55%. In addition, the accuracies for each individual class are calculated as 100% (D0), 96% (D1), 98% (D2), 94% (S2), 92% (S3.0), 94% (S3.1), 52% (S3.2), 66% (S3.3), 92% (F4), 94% (F5) and 96% (F6). Table 5.4 shows the results of the classification. In this table, the columns represent the number of sample images which were manually classified in each category and the rows represent the number of sample

images which were classified into each category by way of the automated method discussed herein. Therefore, the sum of the values in each row is equal to the number of images of each class which was analyzed in the test, and the sum of the values in each column is equal to the number of images detected as each class. Finally, this means that the values in the *(i,j)th* cells where $i \neq j$ represent all of the false positive results (those images classified erroneously), and the values in the *(i,j)th* cells where $i = j$ represent all of the true positive results (those images classified correctly).

From this table, it is apparent that the most prominent location for error in the method in automated damage index estimation corresponds to distinguishing between shear-critical RC columns which have spalled concrete surfaces (S3.2) but also may have other damage such as buckled longitudinal reinforcement (S3.3). Many columns are misclassified as belonging to this damage state (S3.2). This is probably due to the lack of effective means to classifying buckling of reinforcement. However, this is not considered a limitation of the method since the values which are of utmost significance are the associated RC column drift capacities. For shear-critical columns in damage state S3.2 and S3.3, the drift capacity is no different. Therefore, as long as the column response mechanism is classified correctly (shear-critical) and the spalled regions in the shear-critical columns are recognized, it does not affect the result whether or not the buckled longitudinal bars are recognized. Alternatively, when the RC column response mechanism is flexure, the capability of the automated method to indicate the existence of longitudinal bar buckling is

essential. Hence, future research efforts could focus on finding a more effective means to classifying the buckling and fracture of reinforcement.

In addition, the performance of the method in determining the response mechanism was independently measured. For the dataset of 50 damaged RC column images, the accuracy (Equation 5.1), precision (Equation 5.2) and recall (Equation 5.3) were calculated with respect to the RC column response mechanism. In this instance, the values in these equations represent the following: *TP* – the number of images correctly specified (shear-critical columns specified as shear-critical and the same for flexure-critical columns); *FP* – the number of images which should not have been classified as the response mechanism of interest but were classified; and *FN* – the number of images which should have been classified as a specific response mechanism but were either not classified or classified as the opposite response mechanism. These three values were calculated for each response mechanism and the final results yield average precision, recall and accuracy of 70.70%, 61.05% and 79.04%, respectively.

$$\mathrm{Pr}ecision = \frac{TP}{FP + TP} \qquad \text{(Eq. 5.2)}$$

$$\mathrm{Re}call = \frac{TP}{TP + FN} \qquad \text{(Eq. 5.3)}$$

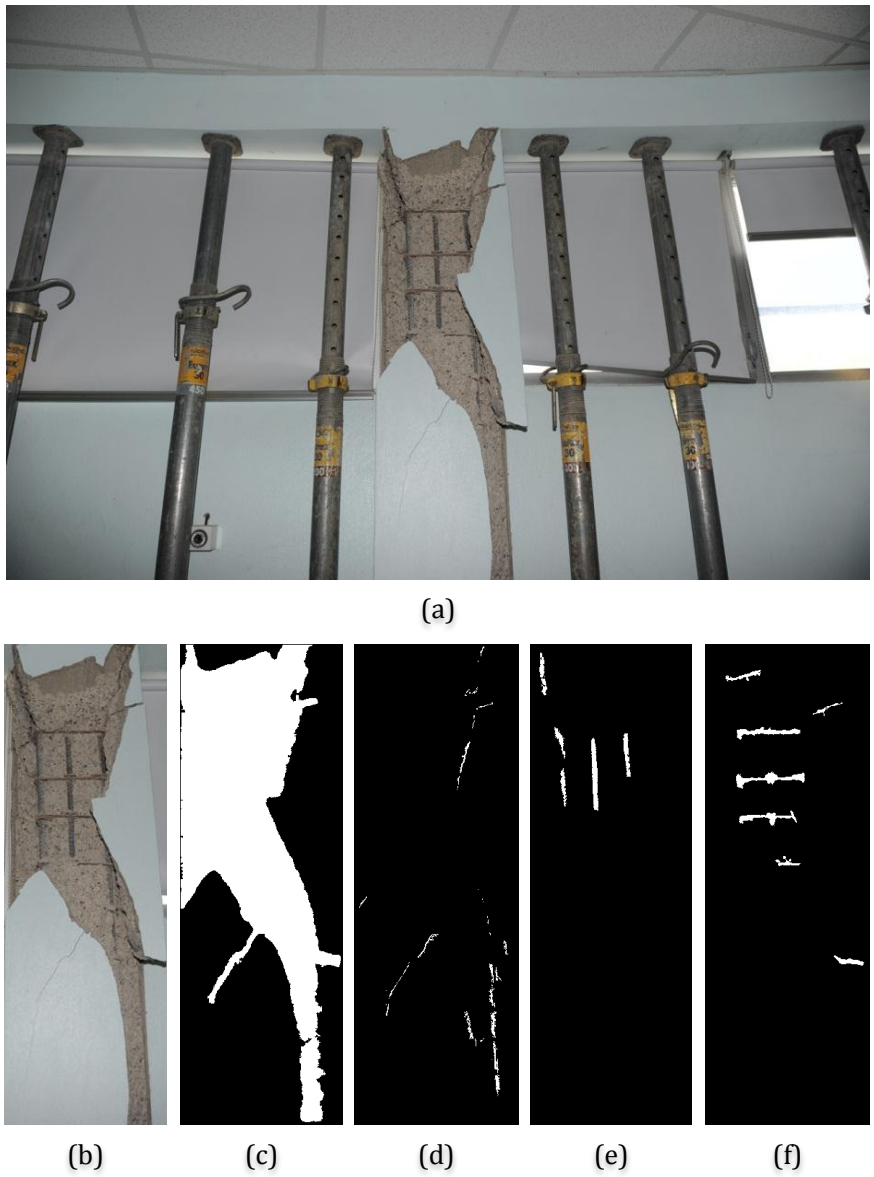


Figure 5.7: Automated detection and property retrieval results: (a) entire image; (b) detected column ROI; (c) spalled ROI; (d) detected cracks; (e) detected longitudinal reinforcement; (f) detected transverse reinforcement

An example of the entire process for automated damage detection, property retrieval and damage index estimation is displayed in Figure 5.7. The column shown in this Figure was evaluated as spalling of the side faces (S3.2) by the manual inspector. By way of the automated damage index estimation procedure, the column

was assumed to have extended beyond S3.2 into S3.3 (longitudinal bar buckling). This example illustrates the existing limitation of the method in automated damage index estimation. Since the constraint for bar buckling is a threshold based on the extent of spalling along the column (L_T), the automated method assumes that the extensive spalled region indicates buckling of reinforcement even when this is not the case. However, as explained previously, the associated drift capacity values are designated as 2.5% (LAL) and 1.5% (HAL). These values are entirely accurate for the RC column pictured here since the associated values are identical for damage states S3.1-S3.4.

5.5. Closure

This chapter presented a novel method in automated damage index estimation of RC columns based only on the observed visible damage. First, the progression of damage on a flexure-critical column was discussed. Maximum drift values on the load vs. drift history were matched with visible damage based on experimental testing, corresponding photographs and textual descriptions of damage observed during testing. Additionally, the progression of damage for a shear-critical RC column was also introduced. Following this, the parameters which were necessary to isolate in these damage progressions were discussed. The necessity to translate the damage states based on the typical damage progression of flexure- and shear-critical columns and the maximum drift values to a set of damage indices which could be related to the residual damage was discussed, and the visual damage indicators which were still viable were presented. Furthermore, the distinction between visual damage observed on each face of the column was presented since

the result of the automated concrete column detection method is a single concrete column surface. The last parameter which was discussed was the column response mechanism (flexure-critical vs. shear-critical). A description of the visual distinction between columns exhibiting these two response mechanisms was provided. Finally, the proposed RC damage indices based on only visible damage and the model for automated damage index estimation were presented. The accuracy of the method in classifying a RC column according to these damage indices was calculated as an overall value of 88.5%. The precision, recall and accuracy for the response mechanism definition were calculated as 70.70%, 61.05% and 79.04%, respectively. These values prove the success of the method in automatically estimating the existing damage state and drift capacity of a RC column based only on the visual damage observed at the residual state.

CHAPTER 6

CONCLUSIONS

In this chapter, first the motivations and objective for the research presented in this dissertation are reviewed. Following this, a brief description of each of the methods created within the context of this research is provided. The chapter will conclude with recommendations and possibilities for future related research in this area.

6.1. Summary and Conclusions

In the event of a natural disaster such as an earthquake, every structure within the affected region is required to be evaluated concerning the safety and remaining structural integrity prior to entry of search and rescue personnel as well as re-entry of the buildings' occupants. The existing procedures for these evaluations rely heavily on the opinion of certified inspectors or structural engineers. The evaluator is required to make a quick assessment of the safety/integrity of the building based primarily on the visual damage which can be observed. These assessment practices are considered unreliable due to the subjective nature and time-consuming. In addition, mobilizing post-earthquake reconainssance teams and assessing damaged buildings, even for a moderately-sized earthquake often take days to weeks to complete. This is the case even for a moderate earthquake.

Prompted by the critical role of post-earthquake inspection in hazard mitigation and the need for its fast performance in earthquake damaged areas, several efforts towards automating building safety assessment have led to the creation of sensing-based evaluation methods. This research proposes an

automated framework for the post-earthquake evaluation of RC buildings using computer vision techniques. The intention is to immediately determine the damage state and maximum drift capacity of reinforced concrete columns in reinforced concrete frame buildings. Under the framework, concrete columns within video data are first detected. Following this, the visible damage inflicted on the detected concrete columns is detected. The spatial properties of the damage are measured in relation to the column's dimensions and orientation. This information is then used to approximate the column's response mechanism and drift capacity. This information is provided in the form of a column damage index which, when supplemented with other building information (structural type and columns arrangement) can be used to query fragility curves of similar buildings, constructed from the analyses of existing and on-going experimental data. Overall, the framework is expected to provide the quantitative assessment of the damage state and increased vulnerability of a reinforced concrete frame in an aftershock. In specific, this research provides a quantitative assessment of the damage state of individual reinforced concrete columns.

In order to support this main objective, the research efforts consisted of four chief activities: (1) automated crack pattern classification; (2) automated spalled region detection and property retrieval; (3) relating reinforced concrete column performance (maximum drift capacity) with visbile, residual damage; and (4) automated reinforced concrete column damage index estimation.

The first method created towards the goal of automated reinforced concrete column safety assessment concerns cracking on the column surfaces. The method is

initiated by retrieving the topological skeleton of concrete crack pixels in the crack map. From this skeleton, the individual crack segments are defined and the crack width, length and orientation are retrieved for each segment. Then, segments are classified according to their orientation, and the spacing between cracks of the same type is calculated. Further, these properties are related with the dimensions of the detected column surface in order to provide relative and meaningful measurements. From this information, the dominant crack pattern on the reinforced concrete column surface is determined.

In addition to cracking, spalling is prevalent in reinforced concrete structures and indicative of varying extents of damage to the member. The second method created in this research is spalled region detection and property retrieval. This method begins by detecting the spalled region (ROI) with a local-entropy based image segmentation algorithm. Then, the extent of the spalling within the ROI on the column surface is evaluated in terms of the length along the longitudinal axis of the column and the depth of the spalling into the column. In order to determine these spalled properties, two novel methods in reinforcement detection are created: one for transverse reinforcement and one for longitudinal reinforcement. The detection of both of these reinforcement types takes place in the CMYK color space which is the dominant color space for steel reinforcement The method in transverse reinforcement detection is principally based on a region-growing operation which compares local pixel intensities and considers the shape and orientation of detected regions before classifying them positively as transverse reinforcement. The method in longitudinal reinforcement detection stems from the characteristic texture of

reinforcing bars and employs a template matching algorithm in each of the C, M, Y and K color channels to locate regions of longitudinal reinforcement within the spalled ROI. With these regions detected, the depth of the spalled region into the column is classified according to the amount and type of reinforcement which is exposed. The length of the spalled region along the column is retrieved by employing a connected component algorithm to determine the length between each detected spalled region and between extreme pieces of exposed transverse reinforcement relative to the column dimensions.

The final method created in this research is the model for automated damage index estimation. This method uses the output from each of the first two methods and translates the damage property information to a quantitative estimate of the existing state of the column: the response mechanism and maximum drift capacity. Before creating the method, the relationship between the visible damage identified by way of the automated methods in damage property retrieval and the maximum drift capacity of the column needed to be defined. The damage progression for two different failure modes (flexure-critical and shear-critical) in reinforced concrete columns had been studied. This relationship between the manner in which damage progresses throughout the cyclic loading of a column was used as a springboard for establishing the relationship between the visual damage observed at the residual state and the maximum drift capacity of the column. Damage states were developed for reinforced concrete columns which took into account the response mechanism, the limited visible damage properties capable of being observed automatically and the general lack of knowledge concerning the column design, material properties,

etc. Following this, the last method included within the scope of this research was created. The damage index and response mechanism for the reinforced concrete column based on a single detected surface was established. First, the method employs the column damage properties to determine which face of the column has been detected. Then, the damage properties are filtered through to determine the response mechanism and finally the maximum drift capacity of the column.

All of the methods proposed in this research have been implemented in the Microsoft Visual Studio .NET environment. The methods have been implemented as an independent and comprehensive prototype, integrating each of the damage type detection and property retrievals with one another. Real images and video data were retrieved from post-earthquake reconnaissance trips in addition to earthquake image databases, and these images formed the database which was used to test the validity of each of these methods as well as the overall method. A manual assessment by visual inspection of the damaged members in these images was performed in order to compare and further assess the validity of the automated methods presented in this research. The performance of each of these methods was evaluated according to several metrics common to computer vision and image processing algorithms. The average measurement error for each of the relative crack properties was estimated as 0.35% (width), 2.21% (length), 6.06% (spacing) and 3.29° (orientation). The average accuracy for the crack classification procedure was calculated as 94.28%. The average measurement error for the relative spalled length measurements was calculated as 4.22% for the length of the spalled region and 8.05% for the length between extreme exposed transverse reinforcement bars.

The average accuracy for the spalled depth retrieval classification procedure was estimated as 87.53%. Then, once employed as input for the automated method in damage index estimation, the precision, recall and accuracy of the methods ability to define the column response mechanism were calculated as 70.7%, 61.05% and 79.04%, respectively. Finally, the accuracy of the overall method in classifying the damage index (and thus, maximum drift capacity) of reinforced concrete columns was calculated as 88.55%.

These results indicated that the damage index for a reinforced concrete column can be estimated automatically with a reasonable amount of error based only on key properties of visibly observed damage on one column surface. If the existing method of concrete column detection employed in this research could be enhanced such that multiple surfaces of the same column were identified as belonging to the same column, this would also improve the method in automated damage index estimation. In addition, the results in automated damage index estimation reveal the added benefit of detecting the existence of core crushing and/or buckling and fracture of longitudinal reinforcement. This would help classify a reinforced concrete column according to the correct damage state with greater accuracy.

6.2. Contributions

The main goal of this research is to help first responders, inspectors and structural specialists make a well-informed decision during infrastructure assessment and rehabilitation tasks in a reliable and automated, thus, rapid manner. The contributions of this research in safety evaluation for emergency response and

structural evaluation for building inspection in the post-earthquake scenario are the following:

1) Reduces the need for mobilizing structural inspectors to assess the safety of buildings after the disaster.
2) Enhances manual visual inspection procedures by providing quantitative inspection results without the reliance on the experience and knowledge of a single inspector or engineer.
3) Alleviates the demand for qualified personnel by providing a suitable alternative/assistant to certified inspectors.
4) Reduces the time-consuming and costly nature of existing lengthy inspection procedures. This will reduce the economic and societal impact after the earthquake.
5) Provides a crude, yet rapid, safety evaluation of entry into damaged structures for emergency responders, reducing the risk of injury to responders.

In addition, the contribution of this research extends beyond the field of post-earthquake structural assessments into the following areas:

- In combination with ground motion and shaking intensity maps for the affected area, the damage states of structural building elements (and ultimately, building fragility) resulting from this work could be employed to aide in prioritization of response task forces.
- Likewise, the methods created in this work for the application area of post-earthquake safety and structural assessments could easily be translated to

aide in the efficiency of routine inspections of buildings, bridges and other critical infrastructure.

6.3. Recommendations for Future Work

There are several potential areas in which the research efforts described here can be extended. A few of these are the following:

- The research efforts discussed in this document have the potential to precede various other efforts in a similar direction. Most directly, in addition to this method in automated damage index estimation for reinforced concrete columns, similar methods could be developed for other structural members (i.e., beams, walls, joints), other structural materials (i.e., steel, wood, masonry) and other structure types (i.e., bridges, dams, levees).
- The method in spalled properties retrieval specifically could be extended to include the detection of spalling into the core of the column. At this point, the damage state estimation would be possible for every potential damage state for each response mechanism.
- In the efforts presented in this work, the focus was not only limited by member type and material, but also by the two response mechanisms. Further study could enhance the current method by providing the user with information regarding whether or not the structure's failure mode is flexure-critical, shear-critical or flexure-shear. In addition, a larger number of images could be used to further validate the research efforts discussed in Chapter 5 for the automated damage index model.

- In Chapter 4, in the sensitivity analysis, it was evident that occlusions were a significant issue in this method. Although this is not a serious constraint for the application area of post-earthquake structural and safety assessments, if the method were to be utilized in other fields such as routine assessments, it would be ideal to improve the method's performance in the face of occlusions. A tracking algorithm could be included such that once a damage type or column is detected, it remains detected, but more information regarding the damage type or column can still be collected. Implementing a tracking algorithm could also aide the method by providing the user with two column surfaces which make up one column, rather than just the single surfaces.
- High-dynamic-range imaging (HDRI or HDR) could be used in order to more accurately represent the range of the image intensities. With this type of imaging, more precise methods in damage detection and property retrieval may be possible.
- Methods in machine learning could also be employed in order to provide a higher level of confidence regarding several of the variables used in the assorted methods in damage detection and property retrieval presented in this work.

APPENDIX A

SPALLED REGION DETECTION CODE

```
namespace Gygax
{

  class SpallingDetection : DamageDetection
  {

    private Image<Bgr, Single> original, FINAL_SPALLED_RESULT;
    private Image<Gray, Single> gray;
    private Image<Gray, Single>[] manuals_S;


    public SpallingDetection(Image<Bgr, Single> original,
    Image<Gray, Single> gray,   ConcreteColumnSurface cs)
    {
       this.original = original;
        this.gray = gray;

        InitializeData();

        int rows = gray.Rows, cols = gray.Cols;

        spalled = gray.Copy();
        spalled = DetectSpalledRegion(gray);
        spalled = findContours(spalled, rows * cols / 500, false,
                  true);
        FindExtentofSpalling(spalled);
        CvInvoke.cvThreshold(spalled, spalled, 128, 255,
                             THRESH.CV_THRESH_BINARY);

        SetColumnSpalledProperties(cs);
    }


    public void SetColumnSpalledProperties(ConcreteColumnSurface
                                                cs)
    {
       List<float> relLS = new List<float>(1);

       for (int i = 0; i < spalled_dims.Count; i++)
       {
        relLS.Add((float)spalled_dims[i][0] /
                             (float)cs.ColumnWidth);
```

```
    }

    cs.spalledRegions.SpalledDims = spalled_dims;
    cs.spalledRegions.RelSpalledLengths = relLS;
    if (cs.spalledRegions.RelSpalledLengths.Count == 0)
     cs.MaxRelSpalledLength = 0;
    else
     cs.MaxRelSpalledLength =
    cs.spalledRegions.RelSpalledLengths[0];

    cs.spalledMap = spalled;
    cs.spalledRegions.spalledClass[0] = 1;
}



public static int SpallHeight { get; set; }
public Image<Gray, Single> spalled { get; set; }
public Image<Gray, byte> spalled_For_Cracks { get; set; }


void InitializeData()
{
 manuals_S = MainWindow.manuals_S;
 FINAL_SPALLED_RESULT = new Image<Bgr,
                        Single>(CvInvoke.cvGetSize(original));
 }


 Image<Gray, Single> DetectSpalledRegion(Image<Gray, Single>
                                         image)
 {
  int nhood_dim = 9;
  int pad_size = nhood_dim / 2;
  Image<Gray, Single> result = image.Copy(),
          spalledWithBorder = new Image<Gray,
          Single>(image.Cols + nhood_dim, image.Rows +
          nhood_dim);
  Point offset = new Point(0, 0);
  MCvScalar sc = new MCvScalar(0.0);

  Image<Gray, Single> nhood_Image = new Image<Gray,
  Single>(nhood_dim, nhood_dim);

  int[] nhood = new int[nhood_dim * nhood_dim];
```

```
  double[,] entropy_Matrix = new double[image.Rows,
                                        image.Cols];
  double avg = 0, entr;
  Padding pad = new Padding(pad_size);
 CvInvoke.cvCopyMakeBorder(result, spalledWithBorder, offset,
 BORDER_TYPE.REPLICATE, sc);

 Image<Gray, Single> newImage = result.CopyBlank();
 double minE = 10000, maxE = 0;

  for (int i = pad_size; i < image.Rows + pad_size; i++)
  {
  for (int j = pad_size; j < image.Cols + pad_size; j++)
  {
      for (int m = 0; m < nhood_dim; m++)
      for (int n = 0; n < nhood_dim; n++)
            nhood_Image[m, n] = spalledWithBorder[i + m -
            pad_size, j + n - pad_size];

      entr = CalculateEntropy(nhood_Image);
         if (entr < minE)
           minE = entr;
         else if (entr > maxE)
           maxE = entr;

      entropy_Matrix[i - pad_size, j - pad_size] = entr;
      newImage[i - pad_size, j - pad_size] = new Gray(entr);
      avg += entr;
 }
 }

 newImage = GetScaling(newImage);
 double T = avg + 10.0;
 for (int i = 0; i < result.Rows; i++)
 for (int j = 0; j < result.Cols; j++)
      if (entropy_Matrix[i, j] < T)
            result[i, j] = new    Gray(0.0);
      else
            result[i, j] = new Gray(255.0);
      return result;
}


public Image<Gray, Single> GetScaling(Image<Gray,Single> img)
{
 double[] minV, maxV;
```

```
 Point[] minL, maxL;
 img.MinMax(out minV, out maxV, out minL, out maxL);
 double m = (255) / (maxV[0] - minV[0]);
 double c = -1 * minV[0] * m;
 img.Mul(m);
 img.Add(new Gray(c));
 return img;
}


double CalculateEntropy(Image<Gray, Single> image)
 {
 double diff = 0;
 double[] probs = new double[511];

 for (int i = 0; i < image.Rows - 1; i++)
 {
 for (int j = 0; j < image.Cols - 2; j++)
 {
      diff = (image[i + 1, j].Intensity - image[i,
                                          j].Intensity);
      if ((diff < 256) && (diff > -256))
           probs[(int)diff + 255] = probs[(int)diff+255]+1;
 }
 }

 double n = probs.Sum();
 double entrop = 0;

 for (int i = 0; i < probs.Length - 1; i++)
 {
      probs[i] = probs[i] / n;
      if (probs[i] != 0)
           entrop = entrop - Math.Log(probs[i], 2);
 }

 return entrop;
}


Image<Gray, Single> EntropyFilt(Image<Gray, Single> image,
CvArray<int> nhood)
{
 Size origSize = image.Size;
 int padSize = (nhood.Size.Height - 1) / 2;
 Image<Gray, Single> result = image.Copy(),
```

```
        spalledWithBorder = new Image<Gray,
        Single>(image.Cols + padSize, image.Rows + padSize);
 Point offset = new Point(0, 0); MCvScalar sc = new
 MCvScalar(0.0);
 Padding pad = new Padding(padSize);
 CvInvoke.cvCopyMakeBorder(result, spalledWithBorder, offset,
 BORDER_TYPE.REPLICATE, sc);
 return result;
}



void FindExtentofSpalling(Image<Gray, Single> image)
{
       this.spalled_dims = FindConnectedComponents(image);
       this.spalled_length = spalled_dims[0][0];
       this.spalled_width = spalled_dims[0][1];
}


List<int[]> FindConnectedComponents(Image<Gray, Single>
                                          dir_im)
{
 int height = dir_im.Rows, width = dir_im.Cols, num_edges= 0;
 int[,] tmpL = new int[height, width];
 List<int> lengths = new List<int>(1), widths = new
                                          List<int>(1);
 List<int> locations = new List<int>(1);
 int LS = 0, WS = 0;
 int[] maxLS = new int[8], maxWS = new int[8];
 List<int>[] edges_vertical_coordinate = new List<int>[5000],
                  edges_horizontal_coordinate = new
                                             List<int>[5000];

 List<int[]> results = new List<int[]>();

 for (int i = 2; i < height - 2; i++)
 {
 for (int j = 2; j < width - 2; j++)
 {
      if (dir_im[i, j].Intensity > 0 && tmpL[i, j] == 0)
      {
            num_edges++;
            tmpL[i, j] = num_edges;
            edges_vertical_coordinate[num_edges - 1] =
                                          new List<int>();
```

```
                    edges_vertical_coordinate[num_edges -
                          1].Add(i);
                    edges_horizontal_coordinate[num_edges - 1] =
                          new List<int>();
                    edges_horizontal_coordinate[num_edges -
                          1].Add(j);
                    Assign_Label(num_edges,
                                 i,
                                 j,
                     edges_vertical_coordinate[num_edges - 1],
                     edges_horizontal_coordinate[num_edges - 1],
                                 dir_im,
                                 tmpL);
          }
}
}

for (int i = 0; i < num_edges; i++)
{
      int max_L = edges_vertical_coordinate[i].Max();
      int min_L = edges_vertical_coordinate[i].Min();
      lengths.Insert(i, max_L - min_L);
      int n = 10;

      while ((min_L + (max_L-min_L)/2.0) <= n*height / 10 &&
                    n > 0)
            n--;

      locations.Insert(i, n);
      int max_W = edges_horizontal_coordinate[i].Max();
      int min_W = edges_horizontal_coordinate[i].Min();
      widths.Insert(i, max_W - min_W);
}
int maxLoc = 0;
int numLSSaved = Math.Min(num_edges, 8);
for (int i = 0; i < numLSSaved; i++)
{
      int[] entry = new int[3];
      maxLS[i] = lengths.Max();
      entry[0] = maxLS[i];
      maxLoc = lengths.IndexOf(maxLS[i]);
      entry[2] = locations[maxLoc];
      lengths.Remove(maxLS[i]);
      maxWS[i] = widths.Max();
      entry[1] = maxWS[i];
      widths.Remove(maxWS[i]);
```

```
        results.Add(entry);
    }


static void Assign_Label(int num_edges, int i, int j,
List<int> edges_x, List<int> edges_y, Image<Gray, Single>
            dir_im, int[,] tmpL)
{
    Stack<int> i_stack = new Stack<int>();
    Stack<int> j_Stack = new Stack<int>();
    Add_To_Stack(i_stack,
                 j_Stack,
                 i,
                 j,
                 dir_im,
                 tmpL);

    while (i_stack.Count() != 0)
    {
        if (tmpL[i_stack.Peek(), j_Stack.Peek()] == 0)
        {
                 tmpL[i_stack.Peek(), j_Stack.Peek()] =
                        num_edges;
                 edges_x.Add(i_stack.Peek());
                 edges_y.Add(j_Stack.Peek());
                 Add_To_Stack(i_stack,
                              j_Stack,
                              i_stack.Pop(),
                              j_Stack.Pop(),
                              dir_im,
                              tmpL);
            }
            else
            {
                i_stack.Pop();
                j_Stack.Pop();
            }
    }
}


static void Add_To_Stack(Stack<int> i_stack, Stack<int>
j_stack, int i, int j, Image<Gray, Single> dir_im, int[,]
tmpL)
{
```

```
if (i > 0 && j > 0 && i < dir_im.Rows - 1 && j < dir_im.Cols
          - 1)
{
     if (dir_im[i - 1, j - 1].Intensity > 0 && tmpL[i - 1,
          j - 1] == 0)
     {
              i_stack.Push(i - 1);
              j_stack.Push(j - 1);
     }
     if (dir_im[i - 1, j].Intensity > 0 && tmpL[i - 1, j]
          == 0)
     {
              i_stack.Push(i - 1);
              j_stack.Push(j);
     }
     if (dir_im[i - 1, j + 1].Intensity > 0 && tmpL[i - 1,
          j + 1] == 0)
     {
              i_stack.Push(i - 1);
              j_stack.Push(j + 1);
     }
     if (dir_im[i, j - 1].Intensity > 0 && tmpL[i, j - 1]
          == 0)
     {
              i_stack.Push(i);
              j_stack.Push(j - 1);
     }
     if (dir_im[i, j + 1].Intensity > 0 && tmpL[i, j + 1]
          == 0)
     {
              i_stack.Push(i);
              j_stack.Push(j + 1);
     }
     if (dir_im[i + 1, j - 1].Intensity > 0 && tmpL[i + 1,
          j - 1] == 0)
     {
              i_stack.Push(i + 1);
              j_stack.Push(j - 1);
     }
     if (dir_im[i + 1, j].Intensity > 0 && tmpL[i + 1, j]
          == 0)
     {
              i_stack.Push(i + 1);
              j_stack.Push(j);
     }
```

```
                if (dir_im[i + 1, j + 1].Intensity > 0 && tmpL[i + 1,
                      j + 1] == 0)
                {
                          i_stack.Push(i + 1);
                          j_stack.Push(j + 1);
                }
            }
          }
        }
}
```

APPENDIX B

DAMAGE PROPERTY RETRIEVAL CODE

```
namespace Gygax
{
    class DamageDetection
    {
        protected Image<Bgr, Single> ORIGINAL, CMYK;
        protected Image<Bgr, Byte> ORIGINAL_B, CMYK_B;
        protected Image<Bgr, Byte> SPALLED, DET_COLUMNS, CRACKED,
                                        REBAR;
        protected Image<Gray, Single> GRAY {get; set;}
        protected static int damage_Type { get; set; }
        protected static float Ts { get; set; }


        public DamageDetection() {}


        public DamageDetection(Image<Bgr, Single> img,
                               ConcreteColumnSurface CS)
        {
            if (img == null) return;
            ORIGINAL = img;
            ORIGINAL_B = new Image<Bgr, byte>(ORIGINAL.Bitmap);
            InitializeData();

            float[] TsVals = { 0.15f, 0.18f, 0.25f };
            CrackDetection cd;
            if (DoesSpallingExist())
            {
                SpallingDetection sd = new
                            SpallingDetection(ORIGINAL, GRAY, CS);

                for (damage_Type = 0; damage_Type < 3;
                      damage_Type++)
                {
                    Ts = TsVals[damage_Type];
                    cd = new CrackDetection(ORIGINAL_B,
                                            damage_Type, CS);
                }
            }
```

```
        else
        {
            SPALLED_REGION = new Image<Gray,
                                       Single>(GRAY.Size);
            CS.spalledMap = SPALLED_REGION;
            CS.spalledRegions.Extent = 0;
            CS.spalledRegions.spalledClass[0] = 0;
            CS.spalledRegions.spalledClass[1] = 0;
            CS.spalledRegions.spalledClass[2] = 0;
            Ts = TsVals[0];
            cd = new CrackDetection(ORIGINAL_B, 0, CS);
        }

        SetColumnGeneralDamageProperties(CS);
    }


    void
SetColumnGeneralDamageProperties(ConcreteColumnSurface cs)
    {
        if (cs.spalledRegions.spalledClass[2] == 1)
        {
            if (cs.spalledRegions.spalledClass[1] == 1)
                cs.spalledRegions.Extent = 4;
            else
                cs.spalledRegions.Extent = 3;
        }

        else if (cs.spalledRegions.spalledClass[1] == 1)
            cs.spalledRegions.Extent = 2;
        else if (cs.spalledRegions.spalledClass[0] == 1)
            cs.spalledRegions.Extent = 1;
    }


    void InitializeData()
    {
            GRAY = new Image<Gray,
                 Single>(CvInvoke.cvGetSize(ORIGINAL));
            GRAY_B = new Image<Gray,
                 Byte>(CvInvoke.cvGetSize(ORIGINAL_B));
      CvInvoke.cvCvtColor(ORIGINAL.Ptr, GRAY.Ptr,
      COLOR_CONVERSION.CV_BGR2GRAY);
      CvInvoke.cvCvtColor(ORIGINAL_B.Ptr, GRAY_B.Ptr,
      COLOR_CONVERSION.CV_BGR2GRAY);
```

```
}
Boolean DoesSpallingExist()
{
    sample = new Sample(GRAY);
    if (sample.Entr < 5.25 || sample.Sdv < 17)
        return false;
    else
        return true;
}


public void FindSpalledLengths()
{
   L_T = 0; L_S = 0;
   List<Point> SP_points = new List<Point>(), TR_points =
   new List<Point>();

   for (int i = 0; i < TR_MAP.Rows; i++)
   {
        for (int j = 0; j < TR_MAP.Cols; j++)
        {
            if (!(TR_MAP[i, j].Equals(new Gray(0.0))))
                TR_points.Add(new Point(j, i));

            if (!(SPALLED_REGION[i, j].Equals(new
                        Gray(0.0))))
                SP_points.Add(new Point(j, i));
        }
    }

    if(TR_points.Count > 0)
        L_T = Math.Abs(TR_points[TR_points.Count - 1].Y -
                        TR_points[0].Y);

    if (SP_points.Count > 0)
        L_S = Math.Abs(SP_points[SP_points.Count - 1].Y -
                        SP_points[0].Y);
}


public float L_T { get; set; }
public float L_S { get; set; }


public bool ReinfRegionLargeEnough(Image<Gray, Single>
reinf, Image<Gray, Single> spalled)
```

```
{
     CountPixels(reinf); int reinfPixels = whitePixels;
     CountPixels(spalled);

     if (reinfPixels > 0.02 * whitePixels)
          return true;
     else
          return false;
 }


 public bool SpalledRegionLargeEnough(Image<Gray,Single>
                                          spalled)
 {
     CountPixels(spalled);
     if (whitePixels > 0.05 * totPixels)
          return true;
     else
          return false;
 }


 public void CountPixels(Image<Gray, Single> image)
 {
     this.whitePixels = 0; this.blackPixels = 0;
     this.totPixels = 0;
     for (int i = 0; i < image.Rows; i++)
         for (int j = 0; j < image.Cols; j++)
         {
             totPixels++;
             if (image[i, j].Equals(new Gray(0.0)))
                         blackPixels++;
             else whitePixels++;
         }
 }

public Image<Gray, Single> ApplyMorphOperations(Image<Gray,
Single> img, int horBound, int vertBound, Boolean open)
 {
    StructuringElementEx SE = new
    StructuringElementEx(horBound, vertBound, horBound /
    2, vertBound / 2, CV_ELEMENT_SHAPE.CV_SHAPE_RECT);
         if (open == true)
         img._MorphologyEx(SE, CV_MORPH_OP.CV_MOP_OPEN,
                                1);
         else
```

```
            img._MorphologyEx(SE, CV_MORPH_OP.CV_MOP_CLOSE,
                                  1);
            return img;
    }


public Image<Bgr, Byte> CompleteDetection(Image<Bgr, Byte>
original, Image<Gray, Single> detected, int showColor)
    {
        float alpha = 0.5f;
        double red, green, blue;
        float showred, showgreen, showblue;

        showred = 0; showgreen = 0; showblue = 0;
        switch (showColor)
        {
            case 0:
                showblue = 255;
                break;
            case 1:
                showgreen = 255;
                break;
            case 2:
                showred = 255;
                break;
            case 3:
                showblue = 100;
                showred = 100;
                break;
        }

        int step_x = 0, step_y = 0;
        if (original.IsROISet)
        {
            step_x = original.ROI.X;
            step_y = original.ROI.Y;
            CvInvoke.cvResetImageROI(original);
        }

        Image<Bgr, Byte> result = original.Copy();
        result.ToBitmap();

        for (int i = 0; i < detected.Rows; i++)
            for (int j = 0; j < detected.Cols; j++)
                if (!(detected[i, j].Equals(new Gray(0.0))))
                {
```

```
                        red = result[i + step_y, j +
                                        step_x].Red;
                        green = result[i + step_y, j +
                                        step_x].Green;
                        blue = result[i + step_y, j +
                                        step_x].Blue;
                    DET_COLUMNS[i + step_y, j + step_x] = new
                    Bgr((blue * (1 - alpha) + alpha *
                    showblue), (green * (1 - alpha) + alpha *
                    showgreen),(red * (1 - alpha) + alpha *
                    showred));
                }

        original.ROI = new Rectangle(new Point(step_x,
                                    step_y), detected.Size);
        return DET_COLUMNS;
    }


public Image<Gray, Single> FillInRegion(Image<Gray, Single>
                        image, int horBound, int vertBound)
    {
        image = ApplyMorphOperations(image, horBound,
                                        vertBound, false);
        image = ApplyMorphOperations(image, horBound,
                                        vertBound, true);
        return image;

    }

public int MaxHeight { get; set; }

public Image<Gray, Single> findContours(Image<Gray, Single>
img, int T_Area, bool justEdges, bool TRandLR)
    {
        Image<Gray, Single> result;

        if (img != null)
        {
            MemStorage pointSt = new MemStorage();
            Image<Gray, byte> canny = new Image<Gray,
                                        byte>(img.Bitmap);
            Image<Bgr, byte> contourImg = new Image<Bgr,
                                        byte>(img.Bitmap);
            result = new Image<Gray, Single>(img.Size);
            bool largeContour = false;
```

```
using (canny)
{
for (Contour<Point> contour =
canny.FindContours(CHAIN_APPROX_METHOD.CV_CHAIN_A
PPROX_SIMPLE, RETR_TYPE.CV_RETR_EXTERNAL,
pointSt);
     contour != null; contour = contour.HNext)
      {
        Rectangle rect =
                 contour.BoundingRectangle;
        MCvBox2D box = contour.GetMinAreaRect();
        int area = rect.Height * rect.Width;

        List<Point> pts = new List<Point>();
        for (int i = 0; i < result.Rows; i++)
        for (int j = 0; j < result.Cols; j++)
        {
            if (justEdges)
            {
                if (contour.InContour(new
                       PointF(j, i)) == 0)
                    pts.Add(new Point(j, i));
            }

            else
            {
                if (contour.InContour(new
                       PointF(j, i)) >= 0)
                    pts.Add(new Point(j, i));
            }
        }

        if (TRandLR)
        {
            if (area > T_Area)
            {
                largeContour = true;
                contourImg.Draw(contour, new
                       Bgr(0, 0, 255), 1);
                for (int nPt = 0; nPt <
                       pts.Count; nPt++)
                    result[pts[nPt]] = new
                       Gray(255);
            }
        }
```

```
                else
                {
                    if ((EstCirc(pts)) && (area >
                            (T_Area)))
                    {
                        largeContour = true;
                        for (int nPt = 0; nPt <
                            pts.Count; nPt++)
                          result[pts[nPt]] = new
                            Gray(255);
                        contourImg.Draw(contour, new
                            Bgr(0, 0, 255), 1);
                    }
                }
            }

            if (!largeContour) result = img.CopyBlank();
        }
    }
    else result = img.CopyBlank();
    return result;
}


// Calculates the squared dot product of 2 points
public double DotProduct(PointF p1, PointF p2)
{
    return (p1.X * p2.X + p1.Y * p2.Y);
}


// Determines if two lines are near-collinear
public double collinear(PointF p1, PointF p2, PointF p3)
{
    PointF d1 = new PointF(p2.X - p1.X, p2.Y - p1.Y);
    PointF d2 = new PointF(p3.X - p1.X, p3.Y - p1.Y);

  if (DotProduct(d1, d2) == 0 || DotProduct(d1, d1) == 0
        || DotProduct(d2, d2) == 0)
   return -1;
    else
    return (Math.Pow(DotProduct(d1, d2), 2) /
    DotProduct(d1, d1) / DotProduct(d2, d2));
}
```

```
// Calculates the squared dot product of 2 points
public double DotProduct(Point p1, Point p2)
{
    return (p1.X * p2.X + p1.Y * p2.Y);
}


// Determines if two lines are near-collinear
public double collinear(Point p1, Point p2, Point p3)
{
    Point d1 = new Point(p2.X - p1.X, p2.Y - p1.Y);
    Point d2 = new Point(p3.X - p1.X, p3.Y - p1.Y);

  if (DotProduct(d1, d2) == 0 || DotProduct(d1, d1) == 0
              || DotProduct(d2, d2) == 0)
        return -1;
    else
    return (Math.Pow(DotProduct(d1, d2), 2) /
   DotProduct(d1, d1) / DotProduct(d2, d2));

}


public Image<Gray, Single> UnspalledROI(Image<Gray,
                                        Single> image)
{
    return image.And(UNSPALLED_REGION);
}


public Image<Gray, Single> SpalledROI(Image<Gray, Single>
                                        image)
{
        if (!SPALLED_REGION.Size.Equals(image.Size))
        image = image.Resize(SPALLED_REGION.Width,
        SPALLED_REGION.Height,INTER.CV_INTER_CUBIC);
        return image.And(SPALLED_REGION);
}

public static double Dist(Point p1, Point p2)
{
    return Dist(p1.X, p1.Y, p2.X, p2.Y);
}
```

```
//Computes Euclidean distance between point (x1,y1) and point
(x2,y2)
        public static double Dist(double x1, double y1, double
                                        x2, double y2)
        {
            return Math.Sqrt(Math.Pow(x2 - x1, 2) + Math.Pow(y2 -
                                    y1, 2));
        }


        protected static Image<Gray, Single> SPALLED_REGION
                        { get; set; }
        protected static Image<Gray, Single> UNSPALLED_REGION
                        { get; set; }
        protected static Image<Gray, Single> TR_MAP { get; set; }
        protected static Image<Gray, Single> LR_MAP { get; set; }
        protected static Image<Gray, Single> CRACK_MAP { get;
                                                        set; }
        public Image<Gray, Single> COLORLESS { get; set; }
        public Sample sample { get; set; }
        public int spalled_length { get; set; }
        public int spalled_width { get; set; }
        public List<int[]> spalled_dims { get; set; }
        public int numCorrectlyDetected { get; set; }
        public double precision { get; set; }
        public double recall { get; set; }
        public double accuracy { get; set; }
        public int totPixels { get; set; }
        public int actualPixels { get; set; }
        public int whitePixels { get; set; }
        public int blackPixels { get; set; }
        public int detPixels { get; set; }
        public double detRatio { get; set; }
    }


        public void TemplateMatchingDetection()
        {
            Image<Gray, Single>[] cmykChannels = cmyk.Split();

            int[,] threshVals = new int[3, 2] {  { 70, 120 },
                                                 { 140, 200 },
                                                 { 140, 200 } };

            int sumZero = 0;
            for (int j = 0; j < ROI.Rows; j++)
```

```
        {
            for (int k = 0; k < ROI.Cols; k++)
            {
                if (ROI[j, k].Equals(new Gray(0.0)))
                        sumZero++;
            }
        }

        for (int i = 0; i < nColorChannels; i++)
        {
            gray = new Image<Gray, Single>(pGrayImg.Bitmap);
            Image<Gray, Single> currentChannel =
                                    gray.CopyBlank();
            currentChannel = cmykChannels[i];

            int Th2 = threshVals[i, 0];
            int Th1 = threshVals[i, 1];
            double[] mean_sdv = findNZAvg(currentChannel);
            double meanval = mean_sdv[0];

            int diff = 0;
            if (i == 0)
                diff = 140 - (int)meanval;
            else
                diff = 210 - (int)meanval;

            Th1 = Th1 - diff;
            Th2 = Th2 - diff;

            threshed = currentChannel.CopyBlank();
            threshed = DoubleThreshold(currentChannel, Th1,
                                            Th2);

            threshed = SpalledROI(threshed);
            ThreshedChannelResults[i] = new Image<Gray,
            Single>(CvInvoke.cvGetSize(pOriginalImg));
            ThreshedChannelResults[i] = threshed;

        }

        FindLongitudinalRebar(ThreshedChannelResults);
    }


    void FindLongitudinalRebar(Image<Gray, Single>[]
                                threshedImgs)
```

```
{
    int horBound = cols / 20, vertBound = rows / 90;
    StructuringElementEx SE1 = new
    StructuringElementEx(horBound, vertBound, horBound /
                         2, vertBound / 2,
    CV_ELEMENT_SHAPE.CV_SHAPE_RECT);

    threshed = CombineResults(threshedImgs);
    for (curr = 0; curr < TOTAL; curr++)
    {
          TemplateMatch(threshed, grTemplates[curr]);
          SingleThreshold(matchedImages[curr]);
          matchedResults[curr] =
          CompleteDetection(matchedImages[curr],
                    grTemplates[curr]);
     }

     LR_Map_BGR = CombineResults(matchedResults, true);
     LR_Map_Gray = gray.CopyBlank();
     ConvertToBW(LR_Map_BGR, LR_Map_Gray);
     LR_Map_Gray = SpalledROI(LR_Map_Gray);

  allRebarAreas = new Image<Gray, byte>(threshed.Size);
  allRebarAreas = new Image<Gray,
  byte>(findContours(threshed, (rows * cols) / 500,
  false, true).Bitmap);
  LR_Map_Gray = ContainThreshedRebar(LR_Map_Gray);
  pCrackMap = new Image<Gray,
  byte>(findContours(LR_Map_Gray, (rows * cols) / 500,
  false, false).Bitmap);
}


public Image<Gray, Single> LR_Map_Gray { get; set; }


void TemplateMatch(Image<Gray, Single> matched, Image<Gray,
Single> template)
{
    matched = matched.MatchTemplate(template,
    TM_TYPE.CV_TM_CCORR_NORMED);
          double min = 0, max = 0;
          Point minLoc = new Point(0, 0);
          Point maxLoc = new Point(0, 0);
    CvInvoke.cvMinMaxLoc(matched.Ptr, ref min, ref max,
    ref minLoc, ref maxLoc, IntPtr.Zero);
```

```
    CvInvoke.cvCvtScale(matched.Ptr, matched.Ptr, 255 /
    (max - min), -255 * min / (max - min));
         matchedImages[curr] = matched;
}


void SingleThreshold(Image<Gray, Single> image)
{
    Gray avg = new Gray(0.0); MCvScalar sdv = new
    MCvScalar(0.0);
    image.AvgSdv(out avg, out sdv);
    double T = avg.Intensity + 2 * sdv.v0;
    double[] avg_sdv = findNZAvgForMatching(image);
    T = avg_sdv[0] + 2* avg_sdv[1];
    CvInvoke.cvThreshold(image.Ptr, image.Ptr, T, 256,
    THRESH.CV_THRESH_BINARY);
}


Image<Gray, Single> CombineResults(Image<Gray, Single>[]
                                        results)
{
    for (int i = 1; i < results.Length; i++)
        results[i] = results[i].And(results[i - 1]);
    return results[results.Length - 1];
}


protected void ThreshCMYK(Image<Bgr, Single> cmyk)
{
    int sumZero = 0;
    int horBound = Math.Max(cols / 50, 2), vertBound =
                    Math.Max(rows / 180, 2);
    Image<Gray, Single>[] ThreshedChannelResults;

    StructuringElementEx SE1 = new
         StructuringElementEx(horBound, vertBound,
         horBound / 2, vertBound / 2,
         CV_ELEMENT_SHAPE.CV_SHAPE_RECT);

    for (int j = 0; j < SPALLED_REGION.Rows; j++)
        for (int k = 0; k < SPALLED_REGION.Cols; k++)
            if (SPALLED_REGION[j, k].Equals(new
                    Gray(0.0)))
                sumZero++;
```

```
if (sumZero == rows * cols)
    m_Threshed = new Image<Gray,
                     byte>(SPALLED_REGION.Bitmap);
else
{
    int num_Channels = cmyk.NumberOfChannels;
    ThreshedChannelResults = new Image<Gray,
                             Single>[num_Channels];
    Image<Gray, Single>[] cmykChannels =
                             cmyk.Split();
    Image<Gray, Single>[] cmykChannelsForTR =
                             cmyk.Split();
 Image<Gray, Byte>[] rebarAreas = new Image<Gray,
 Byte>[num_Channels];
 Image<Gray, Single> threshed;
 int[,] threshVals = new int[3, 2] { { 70, 120 },
 { 140, 200 }, { 140, 200 } };

    for (int i = 0; i < num_Channels; i++)
    {
   Image<Gray, Single> currentChannel = new
   Image<Gray, Single>(pGrayImg.Size);

        currentChannel = cmykChannels[i];

        int Th2 = threshVals[i, 0];
        int Th1 = threshVals[i, 1];
        double[] meansdv = findNZAvg(currentChannel);
        double meanval = meansdv[0];
        int diff = 0;

        if (i == 0)
            if (meanval > 140) diff = 0;
            else diff = 140 - (int)meanval;
        else
            if (meanval > 210) diff = 0;
            else    diff = 210 - (int)meanval;

     Th1 = Th1 - diff;
     Th2 = Th2 - diff;

    Gray avg = new Gray();
    MCvScalar sdv = new MCvScalar();
    currentChannel.AvgSdv(out avg, out sdv);
    Sample sample = new Sample(currentChannel);
    threshed = currentChannel.CopyBlank();
```

```
            threshed = DoubleThreshold(currentChannel, Th1,
            Th2);
            ThreshedChannelResults[i] = new Image<Gray,
            Single>(pGrayImg.Size);
                ThreshedChannelResults[i] = threshed;
            }

        threshed =
        (ThreshedChannelResults[0].Or(ThreshedChannelResults[
        1])).Or(ThreshedChannelResults[2]);
        allRebarAreas = new Image<Gray, byte>(threshed.Size);
        allRebarAreas = new Image<Gray,
        byte>(findContours(threshed, rows * cols / 500,
        false, true).Bitmap);
            allRebarAreas._MorphologyEx(SE1,
                        CV_MORPH_OP.CV_MOP_CLOSE, 1);
            m_Threshed = allRebarAreas.Copy();
            m_Threshed = m_Threshed.SmoothBlur(rows / 30,
                                               rows / 30);
        }
    }


    protected List<Crack> AndRebarThreshedRegion(List<Crack>
                                        cracks, int iter)
    {
        if (iter == 1)
        {
        Image<Bgr, Single> cmyk = new Image<Bgr,
        Single>(pOriginalImg.Bitmap);
        ThreshCMYK(cmyk);
        }

        List<Crack> resultCks = new List<Crack>();
        Crack newCrack;

        for (int nCk = 0; nCk < cracks.Count; nCk++)
        {
            newCrack = new Crack();
            for (int nSeg = 0; nSeg <
                  cracks[nCk].CrackSegments.Count; nSeg++)
            {
                  int numPointsInRegion=0, numPointsTotal=0;
                  foreach (PointF pt in
                              cracks[nCk].CrackSegments[nSeg])
                  {
```

```
                    Point p = new Point((int)Math.Round(pt.X),
                    (int)Math.Round(pt.Y));
                        if (!m_Threshed[p].Equals(new Gray(0.0)))
                        {
                            numPointsInRegion++;
                            numPointsTotal++;
                        }
                        else
                            numPointsTotal++;
                    }
                if (numPointsInRegion > 0.5 *
                cracks[nCk].CrackSegments[nSeg].Count)
                    {

newCrack.CrackSegments.Add(cracks[nCk].CrackSegments[nSeg]);

                        if(iter == 2)

newCrack.SegmentBoxes.Add(cracks[nCk].SegmentBoxes[nSeg]);
                        }
                }

                resultCks.Add(newCrack);
            }

            return resultCks;
        }

    private static Image<Gray, byte> m_Threshed { get; set; }
    private static Image<Gray, byte> allRebarAreas { get; set; }
    int detPixelsT { get; set; }
    int undetPixelsT { get; set; }



    protected void DetectEdges(int N)
        {
            Image<Gray, Single> pBlueChannel =
                                pChannels[nBlueChannel];
            Image<Gray, Single> pGreenChannel =
                                pChannels[nGreenChannel];
            Image<Gray, Single> pRedChannel =
                                pChannels[nRedChannel];

     if (pBlueChannel == null || pGreenChannel == null ||
           pRedChannel == null)
```

```
            return;

Image<Gray, Single> pFirstDerivativeBlueX = new Image<Gray,

 Single>(pBlueChannel.Size);
Image<Gray, Single> pFirstDerivativeBlueY = new Image<Gray,

 Single>(pBlueChannel.Size);

Image<Gray, Single> pFirstDerivativeGreenX = new Image<Gray,

 Single>(pGreenChannel.Size);
Image<Gray, Single> pFirstDerivativeGreenY = new Image<Gray,

 Single>(pGreenChannel.Size);

Image<Gray, Single> pFirstDerivativeRedX = new Image<Gray,

 Single>(pRedChannel.Size);
Image<Gray, Single> pFirstDerivativeRedY = new Image<Gray,

 Single>(pRedChannel.Size);

// Calculate first derivatives in x and y directions from
R,G,B channels
CvInvoke.cvSobel(pBlueChannel, pFirstDerivativeBlueX,1,0,3);
CvInvoke.cvSobel(pBlueChannel, pFirstDerivativeBlueY,0,1,3);
CvInvoke.cvSobel(pGreenChannel,pFirstDerivativeGreenX,1,0,3)
;
CvInvoke.cvSobel(pGreenChannel,pFirstDerivativeGreenY,0,1,3)
;
CvInvoke.cvSobel(pRedChannel, pFirstDerivativeRedX,1, 0, 3);
CvInvoke.cvSobel(pRedChannel, pFirstDerivativeRedY,0, 1, 3);

float[, ,] pFirstDerivativeBlueXData =
                  pFirstDerivativeBlueX.Data;
float[, ,] pFirstDerivativeBlueYData =
                  pFirstDerivativeBlueY.Data;
float[, ,] pFirstDerivativeGreenXData =
                  pFirstDerivativeGreenX.Data;
float[, ,] pFirstDerivativeGreenYData =
                  pFirstDerivativeGreenY.Data;
float[, ,] pFirstDerivativeRedXData =
                  pFirstDerivativeRedX.Data;
float[, ,] pFirstDerivativeRedYData =
                  pFirstDerivativeRedY.Data;
```

```
Image<Gray, Single> pMxx = new Image<Gray,
                           Single>(pBlueChannel.Size);
Image<Gray, Single> pMxy = new Image<Gray,
                           Single>(pBlueChannel.Size);
Image<Gray, Byte> pMyy = new Image<Gray,
                           Byte>(pBlueChannel.Size);

float[, ,] pMxxData = pMxx.Data, pMxyData = pMxy.Data,
           pMyyData = pMxy.Data;

Image<Gray, Single> pV = new Image<Gray,
                           Single>(pBlueChannel.Size);
Image<Gray, Single> pT = new Image<Gray,
                           Single>(pBlueChannel.Size);
float[, ,] pVData = pV.Data, pTData = pT.Data;

int i, j;
int nWidth = pBlueChannel.Width;
int nHeight = pBlueChannel.Height;

// Calculate gradient magnitude and direction of each pixel
for (j = (int)(N / 2); j < nHeight - (int)(N / 2); j++)
for (i = (int)(N / 2); i < nWidth - (int)(N / 2); i++)
{
      pMxxData[j, i, 0] = pFirstDerivativeRedXData[j, i, 0]
                        *pFirstDerivativeRedXData[j, i, 0]
                                    +
                        pFirstDerivativeGreenXData[j, i, 0] *
                        pFirstDerivativeGreenXData[j, i, 0]
                                     +
                        pFirstDerivativeBlueXData[j, i, 0] *
                        pFirstDerivativeBlueXData[j, i, 0];

      pMxyData[j, i, 0] = pFirstDerivativeRedXData[j, i, 0]
                        * pFirstDerivativeRedYData[j, i, 0]
                                          +
                        pFirstDerivativeGreenXData[j, i, 0] *
                        pFirstDerivativeGreenYData[j, i, 0]
                                          +
                        pFirstDerivativeBlueXData[j, i, 0] *
                        pFirstDerivativeBlueYData[j, i, 0];

    pMyyData[j, i, 0] = pFirstDerivativeRedYData[j, i, 0] *
                        pFirstDerivativeRedYData[j, i, 0]
```

```
                                                    +
                pFirstDerivativeGreenYData[j, i, 0] *
                pFirstDerivativeGreenYData[j, i, 0]
                                                    +
                pFirstDerivativeBlueYData[j, i, 0] *
                pFirstDerivativeBlueYData[j, i, 0];
      pVData[j, i, 0] = (float)((Math.Sqrt((pMxxData[j, i,
                           0] + pMyyData[j, i, 0]) *
           (pMxxData[j, i, 0] + pMyyData[j, i, 0]) -
                                        4 * (pMxxData[j,
                    i, 0] * pMyyData[j, i, 0] -
                    pMxyData[j, i, 0] *
                    pMxyData[j, i, 0]))
                    + pMxxData[j, i, 0] + pMyyData[j, i,
                    0]) / 2.0f);

    pTData[j, i, 0] = (float)(Math.Atan2(pVData[j, i, 0] -
                    pMxxData[j, i, 0], pMxyData[j, i,0]));
           }


       MCvScalar scalarMean = CvInvoke.cvAvg(pV,
                           IntPtr.Zero);
       double minValue = 0, maxValue = 0;
       Point minLoc = new Point(0, 0), maxLoc = new Point(0,
                                0);
       CvInvoke.cvMinMaxLoc(pV, ref minValue, ref maxValue,
                           ref minLoc, ref maxLoc,
                           IntPtr.Zero);

       CvInvoke.cvThreshold(pV, pV, 8.0 * scalarMean.v0,
                           maxValue,
                           THRESH.CV_THRESH_TOZERO);

       pMag = pV.Clone();
       pTheta = pT.Clone();
       float[, ,] pEdgeData = pEdges.Data;

       // Find edge pixels
       double[] NeighborDirection = new double[9] { -
                 Math.PI, -0.75 * Math.PI, -0.5 *
                                  Math.PI, -0.25 * Math.PI,

                    0, 0.25 * Math.PI, 0.5 *
                    Math.PI, 0.75 * Math.PI, Math.PI };
```

```
int[] NeighborCandidateX = new int[9] { -1, -1, 0, 1,
                                        1, 1, 0, -1, -1 };
int[] NeighborCandidateY = new int[9] { 0, -1, -1, -
                                        1, 0, 1, 1, 1, 0 };

for (j = 0; j < nHeight; j++)
    for (i = 0; i < nWidth; i++)
    {
        double dThetaData = pTData[j, i, 0];
        double dVData = pVData[j, i, 0];

        if (dVData == 0) continue;

        // Eight directions
        for (int nDirection = 0; nDirection < 9;
               nDirection++)
        {
            double dAngleDiff = Math.Abs(dThetaData -
                    NeighborDirection[nDirection]);

            if (dAngleDiff <= 0.125 * Math.PI)
            {
                int[] NeighborX = new int[2],
                          NeighborY = new int[2];
                double[] dNeighborData = { -99999.0,
                                           -99999.0 };

                NeighborX[0] = i +
                     NeighborCandidateX[nDirection];
                NeighborY[0] = j +
                     NeighborCandidateY[nDirection];

                NeighborX[1] = i -
                     NeighborCandidateX[nDirection];
                NeighborY[1] = j -
                     NeighborCandidateY[nDirection];

                if (NeighborX[0] >= 0 && NeighborX[0]
                          < nWidth && NeighborY[0]
                          >= 0 && NeighborY[0] <
                                nHeight)
                    dNeighborData[0] =
                     pVData[NeighborY[0],
                     NeighborX[0], 0];
```

```
                    if (NeighborX[1] >= 0 && NeighborX[1]
                        < nWidth && NeighborY[1]
                        >= 0 && NeighborY[1] < nHeight)
                        dNeighborData[1] =
                        pVData[NeighborY[1],
                        NeighborX[1], 0];

                    if (dVData > dNeighborData[0] &&
                        dVData > dNeighborData[1])
                        pEdgeData[j, i, 0] = 255;

                }
            }
        }

    for (j = 0; j < nHeight; j++)
        for (i = 0; i < nWidth; i++)
        {
            //if (ROI[j, i].Equals(new Gray(0.0)))
              continue;
            float fEdgeData = pEdgeData[j, i, 0];

            if (fEdgeData == 0) continue;

            int nNeighborEdge = 0;
            double dSumofNeighborEdgeValue = 0;
            double dSumofDiffbtwNeighborEdgeValue = 0;

            for (int nDirection = 0; nDirection < 8;
                    nDirection++)
            {
                int nNeighborX = i +
                    NeighborCandidateX[nDirection];
                int nNeighborY = j +
                    NeighborCandidateY[nDirection];

                if (nNeighborX >= 0 && nNeighborX <
                    nWidth && nNeighborY >= 0 &&
                    nNeighborY < nHeight)
                {
                    if (pEdgeData[nNeighborY, nNeighborX,
                        0] == 255)
                    {
                        nNeighborEdge = 1;
```

```
                                dSumofNeighborEdgeValue +=
                                 pVData[nNeighborY, nNeighborX,
                                                          0];
                                dSumofDiffbtwNeighborEdgeValue +=
                            Math.Abs(pVData[j, i, 0] -
                            pVData[nNeighborY, nNeighborX, 0]);
                             }
                        }

                    }

                    if (dSumofDiffbtwNeighborEdgeValue -
                            dSumofNeighborEdgeValue > 0 ||
                            nNeighborEdge == 0)
                        pEdgeData[j, i, 0] = 0;
                }

            pEdges = new Image<Gray, Single>(pEdgeData);

        } // End DetectEdges method


Image<Gray, Single> DoubleThreshold(Image<Gray, Single> image,
int T1, int T2)
{
        Image<Gray, Single> temp = image.Copy(), result =
                                        image.Copy();
        CvInvoke.cvThreshold(image.Ptr, result.Ptr, T1, 255,
                                THRESH.CV_THRESH_BINARY);
        CvInvoke.cvThreshold(image.Ptr, temp.Ptr, T2, 255,
                                THRESH.CV_THRESH_BINARY);
        CvInvoke.cvAbsDiff(temp.Ptr, result.Ptr, result.Ptr);

        CountPixels(result);
        this.detPixelsT = whitePixels;
        this.undetPixelsT = blackPixels;

         if (detPixelsT > undetPixelsT)
         {
             CvInvoke.cvCvtScale(result.Ptr, result.Ptr, -1,
                                    255);
             int swap = detPixelsT;
             detPixelsT = undetPixelsT; undetPixelsT = swap;
         }

        return result;
```

```
}


public double[] findNZAvgForMatching(Image<Gray, Single>
                                     image)
{
    double[] results = new double[2];

    List<double> nonZeros = new List<double>(1);
    for (int j = 0; j < image.Rows; j++)
    {
        for (int k = 0; k < image.Cols; k++)
        {
            if (image[j, k].Intensity > 10.0)
                nonZeros.Add(image[j, k].Intensity);
        }
    }

    double avg = nonZeros.Average();
    double sdv = 0.0;
    for (int i = 0; i < nonZeros.Count; i++)
        sdv = sdv + Math.Pow((avg - nonZeros[i]), 2.0);

    sdv = Math.Sqrt(sdv / (nonZeros.Count - 1));
    results[0] = avg; results[1] = sdv;
    return results;
}


public double[] findNZAvg(Image<Gray, Single> image)
{
    double[] results = new double[2];

    List<double> nonZeros = new List<double>(1);
    for (int j = 0; j < SPALLED_REGION.Rows; j++)
    {
        for (int k = 0; k < SPALLED_REGION.Cols; k++)
        {
            if (SPALLED_REGION[j, k].Equals(new
                    Gray(0.0)))
                image[j, k] = new Gray(0.0);
            else
                nonZeros.Add(image[j, k].Intensity);
        }
    }
```

```
        double avg = nonZeros.Average();
        double sdv = 0.0;
        for (int i = 0; i < nonZeros.Count; i++)
            sdv = sdv + Math.Pow((avg - nonZeros[i]), 2.0);

        sdv = Math.Sqrt(sdv / (nonZeros.Count - 1));
        results[0] = avg; results[1] = sdv;
        return results;
    }


protected void FindNeighborRebarSegments(List<Crack>
vecCracks, int nCrack, int nCrackSegment, float fSeedOrien,
int[] prev_next)
  {
            int nPrevSegment = -1, nNextSegment = -1;
            float threshDist = fMaxThreshDistance;

            if (nCrack < 0 || nCrack >= vecCracks.Count)
                        return;
       if (nCrackSegment < 0 || nCrackSegment >=
       vecCracks[nCrack].CrackSegments.Count) return;

       PointF seedPoint =
       vecCracks[nCrack].SegmentBoxes[nCrackSegment].center;
       PointF[] seedVerts =
       vecCracks[nCrack].SegmentBoxes[nCrackSegment].GetVerti
       ces();
            float[] seedPN = new float[2];
            float seedOrien = fSeedOrien;

            int nSegment;
            float fPrevDist = 99999, fNextDist = 99999;

       for (nSegment = 0; nSegment <
       vecCracks[nCrack].CrackSegments.Count; nSegment++)
            {
            if (nSegment == nCrackSegment) continue;

         PointF candPoint =
         vecCracks[nCrack].SegmentBoxes[nSegment].center;
         PointF[] candVerts =
         vecCracks[nCrack].SegmentBoxes[nSegment].GetVertices
         ();
            float candOrien =
            vecCracks[nCrack].SegmentBoxes[nSegment].angle;
```

```
        float[] candPN = new float[2];

        if (candOrien < 45 || candOrien > 135)
        {
         candPN[0] =
         Math.Min(Math.Min(Math.Min(candVerts[0].X,
         candVerts[1].X), candVerts[2].X),
         candVerts[3].X);
         candPN[1] =
         Math.Max(Math.Max(Math.Max(candVerts[0].X,
         candVerts[1].X), candVerts[2].X),
         candVerts[3].X);
        }
        else
        {
         candPN[0] =
         Math.Min(Math.Min(Math.Min(candVerts[0].Y,
         candVerts[1].Y), candVerts[2].Y),
         candVerts[3].Y);
         candPN[1] =
         Math.Max(Math.Max(Math.Max(candVerts[0].Y,
         candVerts[1].Y), candVerts[2].Y),
         candVerts[3].Y);
        }

        float fDist = EuclideanDistance(seedPoint,
                                        candPoint);
    float fOrien =
    Math.Abs((float)(Math.Atan2(candPoint.Y -
    seedPoint.Y, candPoint.X - seedPoint.X) * 180.0 /
    Math.PI));

        if (seedOrien < 45 || seedOrien > 135)
        {
         seedPN[0] =
         Math.Min(Math.Min(Math.Min(seedVerts[0].X,
         seedVerts[1].X), seedVerts[2].X),
         seedVerts[3].X);
         seedPN[1] =
         Math.Max(Math.Max(Math.Max(seedVerts[0].X,
         seedVerts[1].X), seedVerts[2].X),
         seedVerts[3].X);

         if (candPoint.X - seedPoint.X > 0 && fDist -
         fNextDist < 0 && seedOrien - (180 - fOrien) < 30)
              {
```

```
                fNextDist = candPN[0] - seedPN[1];
                if (fNextDist < threshDist)
                    nNextSegment = nSegment;
            }

            if (candPoint.X - seedPoint.X <= 0 && fDist -
            fPrevDist < 0 && seedOrien - fOrien < 30)
            {
                fPrevDist = seedPN[0] - candPN[1];
                if (fPrevDist < threshDist)
                    nPrevSegment = nSegment;
            }
        }

        else
        {
        seedPN[0] =
        Math.Min(Math.Min(Math.Min(seedVerts[0].Y,
        seedVerts[1].Y), seedVerts[2].Y),
        seedVerts[3].Y);
        seedPN[1] =
        Math.Max(Math.Max(Math.Max(seedVerts[0].Y,
        seedVerts[1].Y), seedVerts[2].Y),
        seedVerts[3].Y);

        if (candPoint.Y - seedPoint.Y > 0 && fDist -
        fNextDist < 0 && seedOrien - (180 - fOrien) < 30)
            {
                fNextDist = candPN[0] - seedPN[1];
                if (fNextDist < threshDist)
                    nNextSegment = nSegment;
            }
        if (candPoint.Y - seedPoint.Y <= 0 && fDist -
        fPrevDist < 0 && seedOrien - fOrien < 30)
            {
                fPrevDist = seedPN[0] - candPN[1];
                if (fPrevDist < threshDist)
                    nPrevSegment = nSegment;
            }
        }

    } // End nSegment for loop

        prev_next[1] = nNextSegment;
        prev_next[0] = nPrevSegment;
```

```
        } // End FindNeighborRebarSegments method


        void
SetColumnCrackandDamageProperties(ConcreteColumnSurface cs)
        {
            if (DAMAGE_TYPE == 0)
            {
                cs.NumCracksByType = numCrksByType;

                for (int i = 0; i < numCrksByType.Length; i++)
                cs.NumCracksTotal += numCrksByType[i];
                cs.ShearCrackLocs = shearCrackLocs;
                cs.MaxShearCrackWidth = maxShearCrackWidth;
                cs.MaxRelShearCrackWidth = maxShearCrackWidth /
                                           cs.ColumnWidth;
                cs.crackMap = new Image<Gray,
                                        Single>(pCrackMap.Bitmap);
            }

            else if(DAMAGE_TYPE == 1)
            {
                cs.TRMap = new Image<Gray,
                                     Single>(pCrackMap.Bitmap);
                if (ReinfRegionLargeEnough(cs.TRMap,
                                           cs.spalledMap))
                {
                    cs.spalledRegions.spalledClass[1] = 1;
                    cs.spalledRegions.TRLengths = new float[1]
                                                  { length_T };

                 cs.spalledRegions.RelTRLengths.Add(rel_length_T);
                }

                if (cs.spalledRegions.RelTRLengths.Count == 0)
                    cs.MaxRelTRLength = 0;
                else
                    cs.MaxRelTRLength =
                            cs.spalledRegions.RelTRLengths[0];
            }
            else
            {
                cs.LRMap = new Image<Gray,
                                     Single>(pCrackMap.Bitmap);
                if (ReinfRegionLargeEnough(cs.LRMap,
                                           cs.spalledMap))
```

```
                    cs.spalledRegions.spalledClass[2] = 1;
                }

            }


        protected void FindCollinearSegments(List<Crack> vecCracks, int
                        nCrack, int nCrackSegment, float fSeedOrien,
                        double[] collIndices)
            {
                if (nCrack < 0 || nCrack >= vecCracks.Count) return;
                if (nCrackSegment < 0 || nCrackSegment >=
                          vecCracks[nCrack].CrackSegments.Count)
                          return;
                int nSegment;
                MCvBox2D seed =
                    vecCracks[nCrack].SegmentBoxes[nCrackSegment];
                float seedOrien =
                 vecCracks[nCrack].SegmentBoxes[nCrackSegment].angle;

                PointF[] seedVerts = seed.GetVertices();

                PointF seedSz = seed.size.ToPointF();
                PointF seedPtL = new PointF(seed.center.X - seedSz.X
                                            / 2, seed.center.Y -
                                            seedSz.Y / 2);
                PointF seedPtU = new PointF(seed.center.X + seedSz.X
                                            / 2, seed.center.Y +
                                            seedSz.Y / 2);
                seedPtL = seedVerts[0];
                seedPtU = seedVerts[1];
                seedPtL = new PointF((seedVerts[0].X +
                                      seedVerts[3].X) / 2.0f,
                                      (seedVerts[0].Y +
                                      seedVerts[3].Y) / 2.0f);
                seedPtU = new PointF((seedVerts[1].X +
                                      seedVerts[2].X) / 2.0f,
                                      (seedVerts[1].Y +
                                      seedVerts[2].Y) / 2.0f);
                PointF seedPt = seed.center;
                PointF[] candVerts;

                for (nSegment = 0; nSegment <
                    vecCracks[nCrack].CrackSegments.Count;
                                nSegment++)
                {
```

```
                if (nSegment == nCrackSegment) continue;

                PointF candPt =
                 vecCracks[nCrack].SegmentBoxes[nSegment].center;
                candVerts =
        vecCracks[nCrack].SegmentBoxes[nSegment].GetVertices();
                PointF candPtL = new PointF((candVerts[0].X +
                                 candVerts[3].X) / 2.0f,
                                 (candVerts[0].Y +
                                 candVerts[3].Y) / 2.0f);
                PointF candPtU = new PointF((candVerts[1].X +
                                 candVerts[2].X) / 2.0f,
                                 (candVerts[1].Y +
                                 candVerts[2].Y) / 2.0f);

                double h1 = collinear(seedPtL, seedPtU, candPt);
                double h2 = collinear(seedPt, candPtL, candPtU);
                double h = (h1 + h2) / 2.0;
                float candOrien =
                 vecCracks[nCrack].SegmentBoxes[nSegment].angle;
                float fDegreeDiff = Math.Abs(candOrien -
                                                    seedOrien);
                collIndices[nSegment] = h;
                float distance_btw;
                switch(DAMAGE_TYPE)
                {
                    case 1: distance_btw =
                      Math.Max(Math.Abs(candPt.Y - seedPtL.Y),
                      Math.Abs(candPt.Y - seedPtU.Y)); break;
                    case 2: distance_btw =
                      Math.Max(Math.Abs(candPt.X - seedPtL.X),
                      Math.Abs(candPt.X - seedPtU.X)); break;
                    default: distance_btw = 0; break;
                }

                if (h > 0.95 && (fDegreeDiff < 10 || (180 -
                      fDegreeDiff) < 10) &&
                      distance_btw < rows / 30.0)
                    collIndices[nSegment] = 1;
                else if (distance_btw > rows / 30.0)
                    collIndices[nSegment] = collIndices[nSegment]
                                          - 0.1;

            }
        }
}
```

APPENDIX C

REINFORCED CONCRETE COLUMN DAMAGE INDEX ESTIMATION CODE

```
namespace Gygax
{
    class ConcreteColumnDamageIndex
    {
      public List<ConcreteColumnSurface> ConcreteColumn;


       public
       ConcreteColumnDamageIndex(List<ConcreteColumnSurface>
       concreteColumn)
         {
             this.ConcreteColumn = concreteColumn;
         }

         protected int DamageIndex { get; set; }
         protected int ResponseMechanism { get; set; }


         public void EstimateDI()
         {
             RESPONSE_MECH = 2;
             DRIFTS = new float[2];
             for (int i = 0; i < ConcreteColumn.Count; i++)
             {
                 DetectDamage(ConcreteColumn[i]);
                 DetermineColumnFace(ConcreteColumn[i]);
             }

             RetrieveDamageIndex(ConcreteColumn);
             DRIFTS = FindCorrespondingDrifts();
             for(int i = 0; i< ConcreteColumn.Count; i++)
             {
                 ConcreteColumnSurface cs = ConcreteColumn[i];
             }
         }


         protected void DetectDamage(ConcreteColumnSurface
                                          columnSurf)
         {
```

```
            Image<Bgr, Single> columnSurfImg = columnSurf.image;
            DamageDetection dd = new
                DamageDetection(columnSurfImg, columnSurf);
        }
        protected void DetermineColumnFace(ConcreteColumnSurface
                                           columnSurf)
        {

           if (columnSurf.NumCracksByType[0] != 0)  // If any
           flexural cracks on face
                columnSurf.Face = 0;       // Then it is the
           flexural face
           else
                columnSurf.Face = 1;
        }


        protected float[] FindCorrespondingDrifts()
        {
            float[] results = new float[2];

            results[0] =
           MainWindow.ALL_DRIFTS[RESPONSE_MECH][DAMAGE_STATE,
                                                          0];
            results[1] =
           MainWindow.ALL_DRIFTS[RESPONSE_MECH][DAMAGE_STATE,
                                                          1];
            return results;
        }


    protected void RetrieveDamageIndex(List<ConcreteColumnSurface>

      columnSurfaces)
        {
            int iter;

            if (columnSurfaces.Count == 0) return;

            else if (columnSurfaces.Count == 1)
            {
                iter = CheckSpalling(columnSurfaces[0]);
                switch (iter)
                {
                    case 0:
                        CheckCracking(columnSurfaces[0], false);
```

```
                    break;
                case 1:
                    if (columnSurfaces[0].Face == 0)
                    {
                        RESPONSE_MECH = 1; // Shear-critical

            BarBuckling(columnSurfaces[0].spalledRegions);
                    }
                    else
                        CheckCracking(columnSurfaces[0],
                                      true);
                    break;
                case 2:
                    CheckCracking(columnSurfaces[0], true);
                    RESPONSE_MECH = 0; // Flexure-critical

  CheckLongReinforcement(columnSurfaces[0].spalledRegions);
                    break;
            }
        }

        else { }
    }


    protected static int RESPONSE_MECH { get; set; }
    protected static int DAMAGE_STATE { get; set; }
    protected static float[] DRIFTS { get; set; }


protected void CheckCracking(ConcreteColumnSurface columnSurf,
bool spallingExists)
    {
        if (spallingExists)
        {
            if (RESPONSE_MECH == 0)
            {
             if ((columnSurf.NumCracksByType[1] > 0) &&
             (WideShearCracks(columnSurf)))
                  {
                    RESPONSE_MECH = 1;
                    BarBuckling(columnSurf.spalledRegions);
                    return;
                  }
            }
            if (columnSurf.NumCracksByType[1] > 0)
```

```
        {
            RESPONSE_MECH = 1;
            BarBuckling(columnSurf.spalledRegions);
            return;
        }
        else if (columnSurf.NumCracksByType[2] > 0)
        {
            if(WideShearCracks(columnSurf))
            {
                RESPONSE_MECH = 1;
                BarBuckling(columnSurf.spalledRegions);
                return;
            }
        }
        else
        {
            RESPONSE_MECH = 0;

CheckLongReinforcement(columnSurf.spalledRegions);
            return;
        }
 }

 else
 {
     if (columnSurf.NumCracksTotal == 0)
     {
         if (columnSurf.Face == 0) return;
         else { DAMAGE_STATE = 1; return; }
     }
     else
     {
         if (columnSurf.Face == 0) { DAMAGE_STATE = 1;
                                     return; }
         else
         {
             if (columnSurf.NumCracksByType[1] > 0)
            {  RESPONSE_MECH = 1;
               DAMAGE_STATE = 4; return; }
               else
               {
               if (WideShearCracks(columnSurf))
               { RESPONSE_MECH = 1; DAMAGE_STATE = 3;
               return; }
```

```
                else if (MiddleShearCracks(columnSurf))
                { RESPONSE_MECH = 1; DAMAGE_STATE = 2;
                return; }
                      else DAMAGE_STATE = 2;
                  }
              }
          }
      }
}


// 0: No spalling; 1: spalling at top and bottom but no
   exposed LR;
// 2: Spalling at top and bottom w/ exposed LR; 3:
   spalling in middle
protected int CheckSpalling(ConcreteColumnSurface
                            columnSurf)
{
    SpalledRegions sr = columnSurf.spalledRegions;
    if (sr.Extent == 0)
        return 0;

    for (int i = 0; i < sr.SpalledDims.Count; i++)
    {
        if (columnSurf.Face == 0)
        {
            if (sr.SpalledDims[i][2] > 2 &&
                    sr.SpalledDims[i][2] < 7)
                return 1;
        }
        else
        {
            if(sr.SpalledDims[i][2] > 1 &&
                    sr.SpalledDims[i][2] < 8)
                return 1;
        }
    }
    RESPONSE_MECH = 0; // Flexure-critical
    return 2;
}


protected void CheckLongReinforcement(SpalledRegions SR)
{
    if (LRExposed(SR)) BarBuckling(SR);
    Else DAMAGE_STATE = 4;
```

```
        }


        protected bool LRExposed(SpalledRegions SR)
        {
            if (SR.Extent > 2) return true;
            else return false;
        }


        protected void BarBuckling(SpalledRegions SR)
        {
            if (SR.spalledClass[1] == 0) { DAMAGE_STATE = 5;
                                           return; }
            for (int i = 0; i < SR.RelTRLengths.Count; i++)
                if (SR.RelTRLengths[i] > 1 / 2) { DAMAGE_STATE =
                                                  7; return; }
            DAMAGE_STATE = 5;
        }


        protected bool WideShearCracks(ConcreteColumnSurface
                                           columnSurf)
        {
            if(columnSurf.MaxRelShearCrackWidth > 1.0f/60.0f)
                return true;
            else
                return false;
        }


        protected bool MiddleShearCracks(ConcreteColumnSurface
                                           columnSurf)
        {
            for(int i = 0; i < columnSurf.ShearCrackLocs.Count;
                i++)
                if(columnSurf.ShearCrackLocs[i] == 1)
                      return true;
            return false;
        }
    }
}
```

```
//
*****************************************************************
*//
// CLASSES FOR COLUMN SURFACE, CRACK AND SPALLED REGION
      //
//
*****************************************************************
*//

class ConcreteColumnSurface
{
        public ConcreteColumnSurface()
        {
            this.spalledRegions = new SpalledRegions();
            this.NumCracksByType = new int[3];
        }

        public Image<Bgr, Single> image { get; set; }
        public Image<Gray, Single> spalledMap { get; set; }
        public Image<Gray, Single> TRMap { get; set; }
        public Image<Gray, Single> LRMap { get; set; }
        public Image<Gray, Single> crackMap { get; set; }

        public int ColumnHeight { get; set; }
        public int ColumnWidth { get; set; }
        public int Face { get; set; }   // 0 : flexural face, 1 :
                                           side face
        public int DomCrackPattern { get; set; }
       // 0 : flexural, 1 : longitudinal; 2 : shear
        public float MaxRelShearCrackWidth { get; set; }
        public float MaxShearCrackWidth { get; set; }

        public List<int> DomCrackPatterns { get; set; }
       // 0 : flexural, 1 : longitudinal; 2 : shear
        public List<int> ShearCrackLocs { get; set; }
        public int[] NumCracksByType { get; set; } // [flexural
                                           longitudinal shear]
        public int NumCracksTotal { get; set; }
        public SpalledRegions spalledRegions { get; set; }
        public float MaxRelSpalledLength { get; set; }
        public float MaxRelTRLength { get; set; }

    }
```

```
namespace Gygax
{
    class Crack
    {
        public Crack()
        {
            this.CrackSegments = new List<List<Point>>(50);
            this.SegmentBoxes = new List<MCvBox2D>(50);
            this.SegmentOrientations = new List<float>(50);
            this.SegmentLengths = new List<float>(50);
            this.RelSegmentOrientations = new List<float>(50);
            this.RelSegmentLengths = new List<float>(50);
        }

        public float MaxWidth { get; set; }
        public float AvgWidth { get; set; }
        public float RelMaxWidth { get; set; }
       // Max width of crack WRT dimension of structural element
        public float RelAvgWidth { get; set; }
       // Average width of crack WRT dimension of structural
      element
        public Point MaxWidthLoc { get; set; }      // Location
                                                   of maximum width
        public float MaxSegLength { get; set; }
        public List<float> SegmentLengths { get; set; }
        public List<float> SegmentOrientations { get; set; }
        public List<float> RelSegmentLengths { get; set; }
       // Length of crack segments WRT dimension of structural
      element
        public List<float> RelSegmentOrientations { get; set; }
       // Orientation of crack segments WRT dimension of
      structural element

        public List<List<Point>> CrackSegments { get; set; }
        public List<MCvBox2D> SegmentBoxes { get; set; }
        public List<List<double>> CrackOrientations { get; set; }

        public void clearCrackSegments()
        {
            this.CrackSegments = new List<List<Point>>(10);
        }
        public void clearSegmentBoxes()
        {
            this.SegmentBoxes = new List<MCvBox2D>(10);
        }
    }
```

```
}


namespace Gygax
{
    class SpalledRegions
    {
        public SpalledRegions()
        {
            spalledClass = new int[3];
            RelSpalledLengths = new List<float>(1);
            RelTRLengths = new List<float>(1);
        }

        public int Extent { get; set; }     // 0 : No spalling
                                            // 1 : No
                                           reinforcement exposed
                                            // 2 : TR exposed
                                            // 3 : LR exposed
                                            // 4 : Both exposed

        public int[] spalledClass { get; set; }
           // each row holds either a 0 or a 1 if the
           corresponding type of exposure is apparent
           // Row 0 : Cover spalling, Row 1 : TR exposed and Row
           2 : LR exposed
        public List<int[]> SpalledDims { get; set; }
        public float[] TRLengths { get; set; }
                            // [Lt startPt endPt]
        public List<float> RelSpalledLengths { get; set; }
      // [RelLs startPt/(column height) endPt/(column height)]
        public List<float> RelTRLengths { get; set; }
      // [RelLt startPt/(column height) endPt/(column height)]
    }
}
```

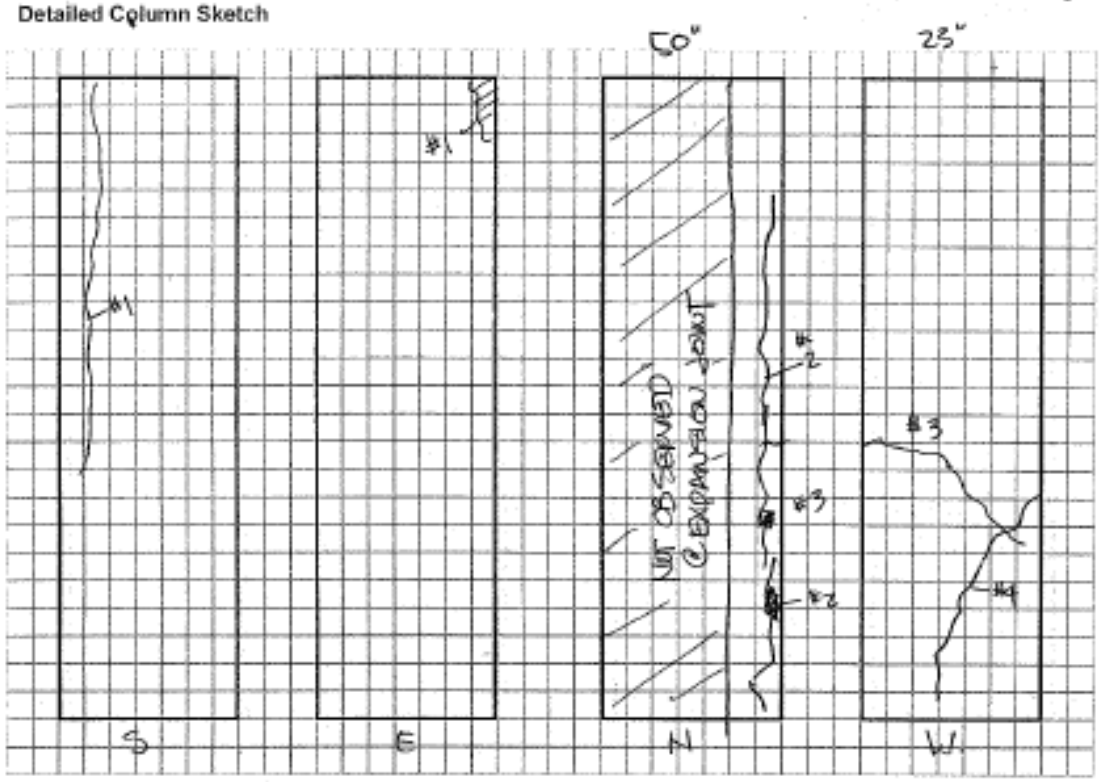
APPENDIX D

MANUAL EVALUATION FORMS

C BLDG I. 7B

~~Learning Center Building~~

Detailed Column Sketch



Component Damage Details

Cracking

Crack #	Width (in)	Length (in)	Orientation	Spacing (in)
1	1/32	75	VERT	—
2	1/32	117	VERT	—
3	1/64	43	5'-40°	—
4	1/32	58	VERT- 50°	—
5				
6				
7				
8				

Probable Response Mode: _____

Additional Notes:
(If applicable, mention here any further exposed details, and make sure to label those details on the sketch as well)

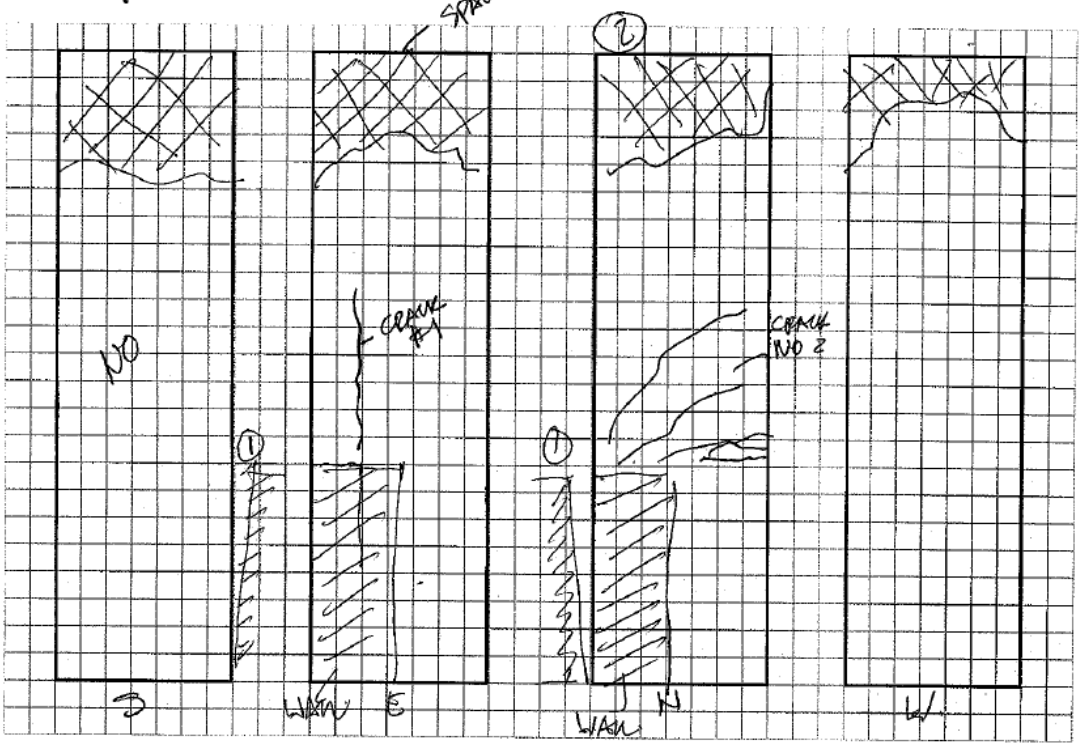
Spalling

Region #	Depth into Member (in)	Transverse Reinforcement Exposed?	Longitudinal Reinforcement Exposed?
1	1"	N	N
2	1/4	N	N
3	3/8	N	N
4			
5			

C PARCEL BLDG 2 A3

Detailed Column Sketch

CITS Building



Component Damage Details

Cracking

Crack #	Width (in)	Length (in)	Orientation	Spacing (in)
1	1/16	3/6	VERT	—
2	1/8	18	HORIZ TO 45°	4 IN.
3				
4				
5				
6				
7				
8				

Spalling

Region #	Depth into Member (in)	Transverse Reinforcement Exposed?	Longitudinal Reinforcement Exposed?
1	FULL	Y	Y
2			
3			
4			
5			

Probable Response Mode: _____

Additional Notes:
(If applicable, mention here any further exposed details, and make sure to label those details on the sketch as well)

① SEPARATED FROM WALL TO EAST

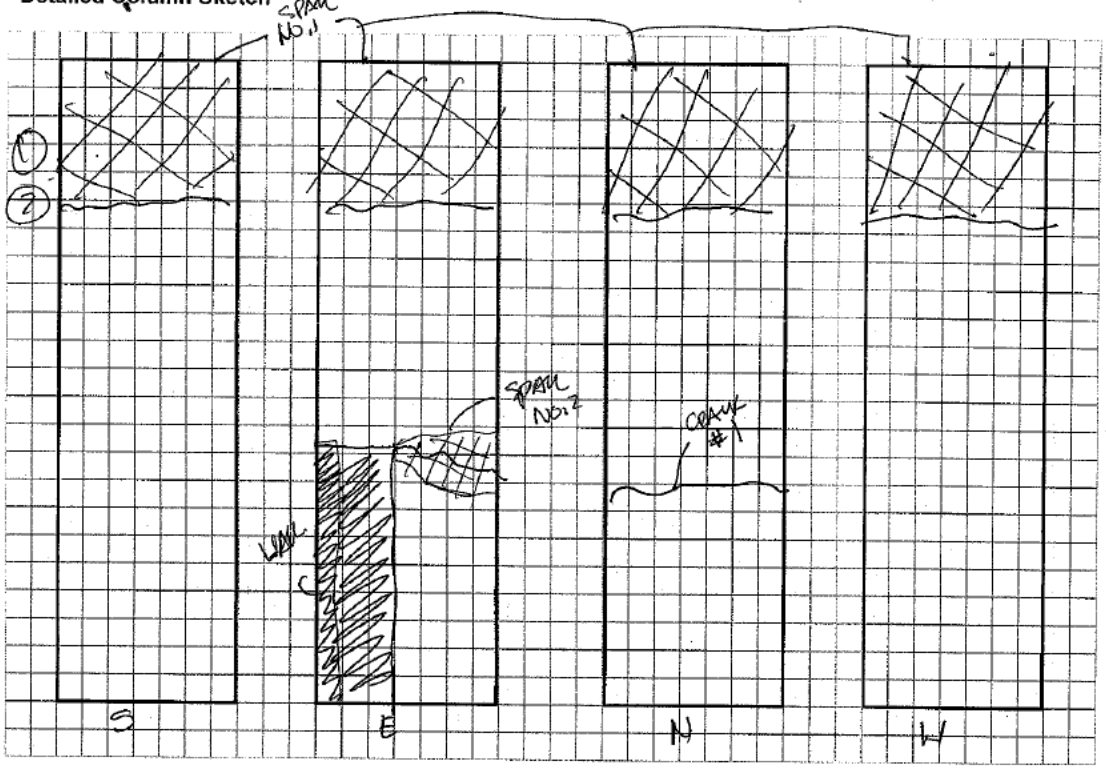
② COLUMN DISPLACED WEST BARS BUCKLED.

CORNER COLUMN

C DIGICEL 2 C8

Detailed Column Sketch

CITS Building



Component Damage Details

Cracking

Crack #	Width (in)	Length (in)	Orientation	Spacing (in)
1	1/32	18	Horiz.	
2				
3				
4				
5				
6				
7				
8				

Spalling

Region #	Depth into Member (in)	Transverse Reinforcement Exposed?	Longitudinal Reinforcement Exposed?
1	FULL 12"	Y	Y
2	1 1/2"	N	N
3			
4			
5			

Probable Response Mode: _____

Additional Notes:
(If applicable, mention here any further exposed details, and make sure to label those details on the sketch as well)

(1) BARS BUCKLED
(2) TIES FAILED - TOP 3.

REFERENCES

Abdel-Qader, I., Abudayyeh, O. and Kelly, M. (2003). Analysis of Edge-Detection Techniques for Crack Identification in Bridges, *Journal of Computing in Civil Engineering, ASCE, 17* (4): 255-263.

Abdel-Qader, I., Pashaie-Rad, S., Abudayyeh, O. and Yehia, S. (2006). PCA-Based Algorithm for Unsupervised Bridge Crack Detection, *Advances in Engineering Software, Elsevier, 37* (12): 771-778.

Adams, B.J. (2004). "Improved disaster management through post-earthquake building damage assessment using multi-temporal satellite imagery," In *Proceedings of the ISPRS XXth Congress, Vol. XXXV,* 12-23 July 2004, Istanbul, Turkey.

Adams, B.J., Mansouri, B. and Huyck, C.K. (2005). Streamlining Post-Earthquake Data Collection and Damage Assessment for the 2003 Bam, Iran, Earthquake Using VIEWS™ (Visualizing Impacts of Earthquakes With Satellites). *Earthquake Spectra, EERI, 21* (S1): S213-S218.

Aldunate, R., Ochoa, S.F., Pena-Mora, F. and Nussbaum, M. (2006). Robust Mobile Ad Hoc Space for Collaboration to Support Disaster Relief Efforts Involving Critical Physical Infrastructure. *Journal of Computing in Civil Engineering, ASCE, 20* (1): 13-27 (2006).

Applied Technology Council (ATC). (1989). ATC-20, "Procedures for Postearthquake Safety Evaluations of Buildings, Report ATC-20," Applied Technology Council. Redwood City, CA.

Applied Technology Council (ATC). (1995). "ATC-20-2, Addendum to the ATC-20 Postearthquake Building Safety Evaluation Procedures," Applied Technology Council. Redwood City, CA.

Applied Technology Council (ATC). (1999) ATC-35, "Earthquake Aftershocks - Entering Damaged Buildings, ATC-35 TechBrief 2," Applied Technology Council. Redwood City, CA.

ASTM. (2008). "ASTM D1654 – 08 Standard Test Method for Evaluation of Painted or Coated Specimens Subjected to Corrosive Environments," West Conshohocken, PA, ASTM International.

Bae, S. (2005). "Seismic Performance of Full-Scale Reinforced Concrete Columns." Ph.D. Thesis, Department of Civil, Architectural and Environmental Engineering, University of Texas at Austin, Austin, TX, 312.

Bearman, C.F. (2012). "Post-Earthquake Assessment of Reinforced Concrete Frames." Master's Thesis, Department of Civil and Environmental Engineering, University of Washington, Seattle, WA, 340.

Bandera, A., Perez-Lorenzo, J., Bandera, J. and Sandoval, F. (2006). Mean shift based clustering of Hough Domain for fast line segment detection, *Pattern Recognition Letters, Elsevier, 27* (2006):578-586.

Brassel, L.D. and Evans, D.D. (2003). Trends in Firefighter Fatalities Due to Structural Collapse, 1979-2002, *National Institute of Standards and Technology,* Fire Research Division, Gaithersburg.

Brilakis, I., Soibelman, L. and Shinagawa, Y. (2006). Construction Site Image Retrieval Based on Material Cluster Recognition, *Journal of Advanced Engineering Informatics, Elsevier, 20* (4):443-452.

Brilakis, I. and Soibelman, L. (2008). Shape-Based Retrieval of Construction Site Photographs, *Journal of Computing in Civil Engineering, ASCE, 22* (1): 14-20.

Cantoni, V., Lombardi, L., Marco, P. and Sicard, N. (2001). "Vanishing point detection: representation analysis and new approaches," In *Proceedings of the 11th International Conference on Image Analysis and Processing,* Pavia, Italy, 90-94.

Celebi, M. (2002). Seismic Instrumentation of Buildings (with Emphasis on Federal Buildings), Technical Report No. 0-7460-68170, United States Geological Survey (USGS), Menlo Park, CA, USA.

Chae, M.J., Iseley, T. and Abraham, D.M. (2003). "Computerized Sewer Pipe Condition Assessment," In *Proceedings of the ASCE International Conference on Pipeline Engineering and Construction,* July 13-16, 2003, Baltimore, MD, 477-493.

Chang, P.C., Flatau, A. and Liu, S.C. (2003). Review Paper: Health Monitoring of Civil Infrastructure, *Journal of Structural Health Monitoring, SAGE, 2* (3): 257-267.

Chen, P.H. and Chang, P.L. (2006). Effectiveness of neuro-fuzzy recognition approach in evaluating steel bridge paint conditions, *Canadian Journal of Civil Engineering, 33* (2006): 103-108.

Chen, P.H., Yang, Y.C. and Chang, L.M. (2010). Box-and-Ellipse-Based ANFIS for Bridge Coating Assessment, *Journal of Computing in Civil Engineering, ASCE, 24* (5): 389-399.

Chen, P.H., Shen, H.K., Lei, C.Y. and Chang, L.M. (2012). Support-vector-machine-based method for automated steel bridge rust assessment, *Automation in Construction, Elsevier, 23* (2012): 9-19.

Cheng, H., Shi, X. and Glazier, C., (2003). Real-time image thresholding based on sample space reduction and interpolation approach, *Journal of Computing in Civil Engineering, ASCE, 17* (4): 264-272.

Chini, M. (2009). Earthquake Damage Mapping Techniques Using SAR and Optical Remote Sensing Satellite Data. *Advances in Geoscience and Remote Sensing, Gary Jedlovec,* ISBN: 978-953-307-005-6.

Chiroiu, L. and Andre, G. (2001). "Damage assessment using high resolution satellite imagery: application to 2001 Bhuj, India earthquake," In *Proceedings of the 7th National Conference on Earthquake Engineering,* EERI.

Chock, G. (2007). "ATC-20 Post-Earthquake Building Safety Evaluations Performed after the October 15, 2006 Hawai'i Earthquakes Summary and Recommendations for Improvements (updated)." Available online: http://www.scd.state.hi.us/HazMitPlan/chapter_6_appM.pdf

Cho, S., Yun, C.-B., Lynch, J.P., Zimmerman, A.T., Spencer Jr., B.F. and Nagayama, T. (2008). Smart Wireless Sensor Technology for Structural Health Monitoring of Civil Structures, *Steel Structures 8* (2008): 267-275.

Collins, S. (2011). "Insurance Too Little for Many Firms to Rebuild," *The New Zealand Herald.* Christchurch, New Zealand. Available online: http://www.nzherald.co.nz/christchurch-earthquake/news/article.cfm?c_id=1502981&objectid=10722381

Cornelis, N. and Van-Gool, L. (2008). "Fast Scale Invariant Feature Detection and Matching on Programmable Graphics Hardware," In *Proceedings of the Computer Vision and Pattern Recognition Workshop, 2008, IEEE Computer Society Conference,* Anchorage, AK.

Cychosz, J. (1994). "Efficient Binary Image Thinning using Neighbourhood Maps," Academic Press Graphics Gems Series-Graphics Gem IV, ISBN 0-12-336155-9, 465-473.

Dai, F., Dong, S., Kamat, V.R. and Lu, M. (2011). Photogrammetry Assisted Measurement of Interstory Drift for Rapid Post-Disaster Building Damage Reconnaissanc, *Journal of Nondestructive Evaluation, Springer, 30* (3): 201-212.

DesRoches, R., Comerio, M., Eberhard, M., Mooney, W. and Rix, G.J. (2011). Overview of the 2010 Haiti Earthquake, *Earthquake Spectra, EERI, 27* (S1): S1-S21.

Dong, P. and Guo, H. (2012). A framework for automated assessment of post-earthquake building damage using geospatial data, *International Journal of Remote Sensing, Taylor and Francis, 33* (1): 81-100.

Eberhard, M.O., Baldridge, S., Marshall, J., Mooney, W. and Rix, G.J. (2010). *The Mw 7.0 Haiti Earthquake of January 12, 2010: USGS/EERI Advance Reconnaissance Team Report.* U.S. Geological Survey Open-File Report, 58.

Elnashai, A.S., Jefferson, T., Fiedrich, F., Cleveland, L.J. and Gress, T. (2009). Impact of New Madrid Seismic Zone Earthquakes on the Central USA, *New Madrid Seismic Zone Catastrophic Earthquake Response Planning Project, MAE Center Report No. 09-03, Volume 1*, Mid-America Earthquake Center.

Fabbri, R., Costa, L., Torelli, J. and Bruno, O. (2008). 2D Euclidean distance transform algorithms: a comparative survey, *ACM Computing Surveys, Association for Computing Machinery (ACM), 40* (1), 2:1-2:44.

Farrar, C.R. (2001). "Historical Overview of Structural Health Monitoring," *Lecture Notes on Structural Health Monitoring Using Statistical Pattern Recognition,* Los Alamos Dynamics, Los Alamos, NM, USA.

Federal Emergency Management Agency (FEMA). (2006). "National Urban Search and Rescue Response System – Structure Specialist Position Description." Available online: http://www.disasterengineer.org/library/Struct%20Spec%20PD%20July%202006.pdf

Federal Emergency Management Agency. (2008). *Structures Specialist Position Description,* DisasterEngineer.org - US&R Structures Specialist Resource. Available online: http://www.disasterengineer.org/LinkClick.aspx?fileticket=7LaKP4hU0jU%3d&tabid=57&mid=397 (Sept, 2012).

Federal Emergency Management Agency (FEMA). (2009a). *Module 1C Structural Engineering Systems – Parts 1 & 2, Structural Collapse Technician Course – Student Manual,* Retrieved from Federal Emergency Management Agency.

Federal Emergency Management Agency (FEMA). (2009b). *Module 1C Structural Engineering Systems - Part 3*, Retrieved from Federal Emergency Management Agency.

Fernandes, L. and Oliveira, M. (2008). Real-time line detection through an improved Hough transform voting scheme, *Pattern Recognition, Elsevier, 41* (1): 299-314.

Field, E. H., Dawson, T. E., Felzer, K. R., Frankel, A. D., Gupta, V., Jordan, T. H., Parsons, T., Peterson, M. D., Stein, R. S., Weldon II, R. J. and Wills, C. J. (2008). "The Uniform California Earthquake Rupture Forecast, Version 2 (UCERF 2)," *USGS Open File Report 2007-1437,* Working Group on California Earthquake Probabilities. Available online: http://pubs.usgs.gov/of/2007/1437/ (Feb, 2013).

German, S., Brilakis, I. and DesRoches, R. (2011). "Automated Detection of Exposed Reinforcement in Post-Earthquake Safety and Structural Evaluations," In *Proceedings of the Modern Methods and Advances in Structural Engineering and Construction Conference,* 21-26 June 2011, Zurich, Switzerland.

German, S., Brilakis, I. and DesRoches, R. (2012). Rapid Entropy-Based Detection and Properties Measurement of Concrete Spalling with Machine Vision for Post-Earthquake Safety Assessments, *Advanced Engineering Informatics, Elsevier, 26* (4): 846-858.

Guo, H., Li, X. and Zhang, L. (2009). Study of detecting method with advanced airborne and spaceborne synthetic aperture radar data for collapsed urban buildings from the Wenchuan earthquake, *Journal of Applied Remote Sensing, SPIE,* 3: 2-19.

Guru, D., Shekar, B. and Nagabhushan, P. (2004). A simple and robust line detection algorithm based on small eigenvalue analysis, *Pattern Recognition Letters, Elsevier, 25* (2004): 1-13.

Hasegawa, H., Yamazaki, F., Matsuoka, M. and Seikimoto, I. (2000). "Determination of building damage due to earthquakes using aerial television images," In *Proceedings of the 12th World Conference on Earthquake Engineering,* Auckland, NZ.

Hashitera, S., Kohiyama, M., Maki, N. and Fujita, H. (1999). "Use of DMSP-OLS images for early identification of impacted areas due to the 1999 Marmara earthquake disaster," In *Proceedings of the 20th Asian Conference on Remote Sensing,* Hong Kong, 1291-1296.

Hipley, P. (2001)."Caltran's Current State of Practice," In *Proceedings of the Instrumental Systems for Diagnostics of Seismic Response of Bridges and Dams,* Consortium of Organizations for Strong-Motion Observation Systems.

Huyck, C.K., Mansouri, B., Eguchi, R.T., Houshmand, B., Castner, L.L. and Shinozuka, M. (2002). "Earthquake damage detection algorithms using optical and ERS-SAR satellite data – application to the August 17, 1999 Marmara, Turkey earthquake," In *Proceedings of the 7th National US Conference on Earthquake Engineering,* 21-25 July 2002, Boston, MA.

Iyer, S. and Sinha, S.K. (2006). Segmentation of pipe for crack detection in buried sewers, *Computer-Aided Civil and Infrastructure Engineering, Wiley, 21*: 395–410.

Kamat, V.R. and El-Tawil, S. (2007). Evaluation of Augmented Reality for Rapid Assessment of Earthquake-Induced Building Damage, *Journal of Computing in Civil Engineering, ASCE, 21* (5): 303-310.

Katz, J.M. (2010). Medals for Haiti Recovery, Little for Homeless, *The Associated Press.*

Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: A More Distinctive Representation for Local Image Descriptors, *Advances in Computer Vision and Pattern Recognition, Springer, 2* (27): II-506-II513.

Kohiyama, M., Hayashi, H., Maki, N., Higashida, M., Kroehl, H.W., Elvidge, C.D. and Hobson, V.R. (2004). Early damaged area estimation system using DMSP-OLS night-time imagery, *International Journal of Remote Sensing, Taylor and Francis, 25* (11): 2015-2036.

Kosecka, J. and Zhang, W. (2002). "Video compass," In *European Conference on Computer Vision,* 657-673.

Kostoulas, D., Aldunate, R., Peña-Mora, F. and Lakhera, S. (2006). "A Decentralized Trust Model to Reduce Information Unreliability in Complex Disaster Relief Operations," In *Proc. 13th EG-ICE Workshop on Intelligent Computing in Engineering and Architecture,* Ascona, Switzerland, June 25-30, 2006, LNCS 4200, 383-407.

Kottapalli, V.A., Kiremidjian, A.S., Lynch, J.P., Carryer, E., Kenny, T.W., Law, K.H. and Lei, Y. (2003). "Two-Tiered Wireless Sensor Network Architecture for Structural Health Monitoring," In *SPIE's 10th Annual International Symposium on Smart Structures and Materials (Vol. 5057),* 8-19, San Diego, CA, 2003.

Lazebnik, S., Schmid, C. and Ponce, J. (2004). "Semi-Local Affine Parts for Object Recognition," In *Proceedings of the British Machine Vision Conference, Vol. 2,* Kingston, UK, September 2004, 959-968.

Lee, S., Chang, L.M. and Skibniewski, M. (2006). Automated recognition of surface defects using digital color image processing, *Automation in Construction, Elsevier, 15* (2006): 540-549.

Lee, S. (2010). An eigenvalue-based recognition method and its application to bridge coating, *Innovation in Architecture Engineering and Construction*.

Lehman, D., Moehle, J., Mahin, S., Calderone, A. and Henry, L. (2004). Experimental Evaluation of the Seismic Performance of Reinforced Concrete Bridge Columns, *Journal of Structural Engineering, ASCE, 130* (6): 869-879.

Leung, T. and Malik, J. (2001). Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons, *International Journal of Computer Vision, Springer, 43* (1): 29-44.

Liu, Z., Shahrel, A., Ohashi, T. and Toshiaki, E. (2002). "Tunnel crack detection and classification system based on image processing," In *Proc. SPIE Vol. 4664, Machine Vision Applications in Industrial Inspection X,* San Diego, CA, USA, 145-152.

Liu, T., Moore, A., Gray, A. and Yang, K. (2004). "An investigation of practical approximate nearest neighbour algorithms," In *Proceedings of Neural Information Processing Systems,* Vancouver, BC, Canada, 825-832.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60* (2): 91-110.

Lukins, T.C. and Trucco, E. (2007). "Towards Automated Visual Assessment of Progress in Construction Projects," In *Proceedings of the British Machine Vision Conference,* Warwick, UK.

Lynch, J.P. (2007). An Overview of Wireless Structural Health Monitoring for Civil Structures, *Philosophical Transactions of the Royal Society A, Mathematical, Physical & Engineering Sciences, 365*: 345-372.

Masumoto, J., Hori, M., Sato, Y., Murakami, T., Johkoh, T., Nakamura, H. and Tamura, S. (2000). Automated detection of liver tumors in X-ray CT images, *IEICE Transactions on Information and Systems, IEICE, J83-D* (1): 219-227.

Matsuoka, M. and Yamazaki, F. (2002). "Application of the damage detection method using SAR intensity images to recent earthquakes," In *Proceedings of the IGARSS,* 24-28 June 2002, Toronto, ON.

McEntire, D. A. and Cope, J. (2004). "Damage Assessment After the Paso Robles (San Simeon, California) Earthquake: Lessons for Emergency Management," Paso Robles.

Micusik, B., Wildenauer, H. and Kosecka, J. (2008). "Detection and matching of rectilinear structures," In *Proceedings of the Computer Vision and Pattern Recognition Workshop, 2008, IEEE Computer Society Conference,* Anchorage, AK.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27* (10): 1615-1630.

Mitomi, H., Matsuoka, M. and Yamazaki, F. (2002). "Application of Automated Damage Detection of Buildings by Panchromatic Television Images," In *Proceedings of the 7th National US Conference on Earthquake Engineering,* Boston, MA.

Murai, S. (1974). Remote Sensing Notes, Chapter 3: Microwave Remote Sensing, *Japan Association on Remote Sensing (JARS),* 56-79.

Nagayama, T. and Spencer Jr., B.F. (2007). Structural Health Monitoring Using Smart Sensors, NSEL Report Series: Report No. NSEL-001, November, 2007.

Neto, J. and Arditi, D. (2002). Using Colors to Detect Structural Components in Digital Pictures, *Computer Aided Civil and Infrastructure Engineering, Wiley, 17* (2002): 61-76.

Network for Earthquake Engineering Simulation (NEES). (2009). NEEShub Databases. Available online: http://nees/org/databases

NIOSH. (2007). "Fire Fighter Fatality Investigation and Prevention Program." Available online:
The National Institute for Occupational Safety and Health | CDC: http://www.cdc.gov/niosh/fire/

Nister, D. and Stewenius, H. (2006). "Scalable Recognition with a Vocabulary Tree," In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2,* 2161-2168.

Ogawa, N. and Yamazaki, F. (2000). "Photo-interpretation of buildings damage due to earthquakes using aerial photographs," In *Proceedings of the 12th World Conference on Earthquake Engineering,* Auckland, NZ.

Open CV. (2010). "Emgu CV: OpenCV in .NET (C#, VB, C++ and More)." Available online: http://www.emgu.com/wiki/index.php/Main_Page (Oct, 2011).

Pacific Earthquake Engineering Research (PEER). (2011). *NISEE Earthquake Engineering Online Archive.* Available online: http://berkeley.edu/elibrary/

Pandey, A.K., Biswas, M. and Samman, M.M. (1991). Damage detection from changes in curvature mode shapes, *Journal of Sound and Vibration, Elsevier, 145* (2): 321-332.

Peña-Mora, F. and Mehta, S. (2009). "Expediting Assessment of Damaged Buildings during Disaster Response Using Mobile Ad Hoc Networks," In *Proceedings of the ASCE International Workshop on Computing in Civil Engineering*, 217-226.

Radio New Zealand (RNZ). (2011). "Painstaking work continues at devastated buildings," 4 March 2011. Available online: http://www.radionz.co.nz/news/canterbury-earthquake/69902/painstaking-work-continues-at-devastated-buildings

Rother, C. (2000). "A new approach for vanishing point detection in architectural environment," In *Proceedings of the 11th British Machine Vision Conference,* Bristol, UK, 11-14 September 2000.

Saito, K., Spence, R.J., Going, C. and Markus, M. (2004). Using high-resolution satellite images for post-earthquake building damage assessment: a study following the 26.1.01 Gujarat earthquake, *Earthquake Spectra, EERI, 20* (1): 145-169.

Schmid, C. (2001). “Constructing models for content-based image retrieval,” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Volume 2*, 39-45.

Sedra, N., Eberhard, M., Irfanoglu, A., Matamoros, A., Pujol, S., Haraldsson, O.S., Lattanzi, D.A., Lauer, S.L., Lyon, B., Messmer, J., Nasi, K., Rautenberg, J., Symithe, S. and Douilly, R. (2010). "The Haiti Earthquake Database." Available online: http://nees.org/resources/1797

Sezen, H. (2002). “Seismic Behavior and Modeling of Reinforced Concrete Building Columns.” Ph.D. Thesis, Dept. of Civil and Environmental Engineering, Univ. of California at Berkeley, Berkeley, CA, 345.

Sezen, H. and Moehle, J.P. (2004). Shear Strength Model for Lightly Reinforced Concrete Columns, *Journal of Structural Engineering, ASCE, 130* (11): 1692-1703.

Shannon, C.E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal, 23*: 379-423.

Shaw, D. and Barnes, N. (2006). “Perspective Rectangle Detection,” In *Applications of Computer Vision, Workshop at the European Conference on Computer Vision,* May 2006.

Silpa-Anan, C. and Hartley, R. (2008). “Optimized KD-trees for fast image descriptor matching,” In *Proceedings of the Computer Vision and Pattern Recognition Workshop, 2008, IEEE Computer Society Conference,* Anchorage, AK.

Simcox, C. (2011). “Thousands of Christchurch Homes Face Demolition,” The Dominion Post, 7 March 2011, Available online: http://www.stuff.co.nz/national/christchurch-earthquake/4740358/Thousands-of-earthquake-damaged-Christchurch-homes-face-demolition

Sinha, S.K. and Fieguth, P.W. (2006). Automated detection of cracks in buried concrete pipe images, *Journal of Automation in Construction, Elsevier, 15* (1): 58-72.

Texas Engineering Extension Service (TEEX). (2007). *Advanced Structural Collapse 4 - Program of Instruction.* Retrieved September 19, 2009, from TEEX: Texas Engineering Extension Service. Available online: http://www.teex.com/USAR/documents/Advanced%20Structural%20Collapse%204%20-%20Program%20of%20Instruction%20-%20Condensed.pdf

The Economist Group. (2011). "Counting the cost," *The Economist online,* 31 March, 2011. Available online: http://www.economist.com/blogs/dailychart/2011/03/natural_disasters (Jan, 2013).

Toksoy, T. and Aktan, A. E. (1994). Bridge-condition assessment by modal flexibility, *Experimental Mechanics, 34,* 271-278.

United Kingdom Fire Service Search and Rescue Team (UKFSSART). (2007). "Structure Collapse - A Guide for Emergency Personnel." Available online: http://www.ukfssart.org.uk/index.htm (Jan, 2009).

United States Fire Administration (USFA). (1994). "Search and Rescue Operations Following the Northridge Earthquake," Los Angeles: Federal Emergency Management Agency.

United States Geological Survey (USGS). (2006). *Earthquake Hazards – A National Threat,* Washington, D.C.: U.S. Department of the Interior.

United States Geological Survey (USGS). (2008). "Forecasting California's Earthquakes—What Can We Expect in the Next 30 Years?" *Fact Sheet 2008-3027,* Washington, D.C.: U.S. Department of the Interior.

United States Geological Survey (USGS). (2009). "Earthquake Hazard in the New Madrid Seismic Zone Remains a Concern," *Fact Sheet 2009-3071,* Washington, D.C.: U.S. Department of the Interior.

United States Geological Survey (USGS). (2012). "Earthquake Facts and Statistics Graphs," *Earthquake Hazards Program,* Washington, D.C.: U.S. Department of the Interior.

Van Erkel, A.R. and Pattynama, P.M. (1998). Receiver operating characteristic (ROC) analysis: basic principles and applications in radiology, *European Journal of Radiology, 27* (2): 88-94.

Van Rijsbergen, C.J. (1979). *Information Retrieval*, 2nd Ed. Butterworth-Heinemann: London.

Varma, M. and Zisserman, A. (2005). A statistical approach to texture classification from single images, *International Journal of Computer Vision 62* (1-2): 61-81.

Wang, C., Zhang, H., Wu, F., Zhang, B., Tang, X., Wu, H., Wen, X. and Yan, D. (2009). Disaster phenomena of Wenchuan earthquake in high resolution airborne synthetic aperture radar images, *Journal of Applied Remote Sensing, 3*: 20-35.

Watson, S. (1989). "Design of Reinforced Concrete Frames of Limited Ductility," Ph.D. Thesis, University of Canterbury, Christchurch, 232.

Yamaguchi, T. and Hashimoto, S., (2010). Fast crack detection method for large-size concrete surface images using percolation-based image processing, *Machine Vision and Applications, Springer, 21* (5): 797-809.

Yu, S., Jang, J. and Han, C. (2007). Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel, *Journal of Computing in Civil Engineering, ASCE, 17* (4): 255-263.

Yusuf, Y., Matsuoka, M. and Yamazaki, F. (2001). "Damage detection from Landsat 7 satellite images for the 2001 Gujurat, India earthquake," In *Proceedings of the 22nd Asian Conference on Remote Sensing,* Singapore.

Yusuf, Y., Matsuoka, M. and Yamazaki, F. (2002). "Detection of building damage due to the 2001 Gujurat, India Earthquake using satellite remote sensing," In *Proceedings of the 7th National US Conference on Earthquake Engineering,* 21-25 July 2002, Boston, MA.

Zhang, Z. and Aktan, A.E. (1995). "The damage indices for constructed facilities," In *Proceedings of the 13th International Modal Analysis Conference,* 1520-1529.

Zhu, Z. and Brilakis, I. (2010). Concrete Column Recognition in Images and Videos, *Journal of Computing in Civil Engineering, ASCE, 24* (6): 478 – 487.

Zhu, Z., German, S. and Brilakis, I. (2011). Visual Retrieval of Concrete Crack Properties for Automated Post-Earthquake Structural Safety Evaluation, *Journal of Automation in Construction, Elsevier, 20* (7): 874-883.

VITA

STEPHANIE ANN GERMAN

Stephanie German was born on June 9, 1987 in Houston, Texas where she grew up. Upon graduating high school in 2005, she attended The University of Texas at Austin. She obtained her Bachelors of Science degree in Architectural Engineering in 2009. Upon completion of her Bachelor's degree, she continued to graduate school at the Georgia Institute of Technology where she first pursued a Master's degree in Civil Engineering. Having obtained her Master's degree in 2011, she began her pursuit of Doctoral studies in Civil Engineering at the Georgia Institute of Technology with a special emphasis on Earthquake Engineering and a minor in Computer Vision.